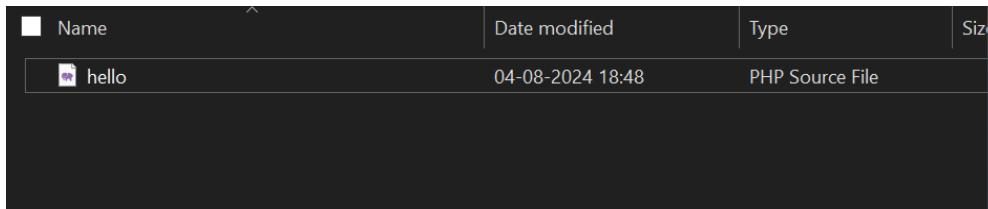


Adv DevOps Experiments

Exp 1a : Static Hosting

a) Hosting of a PHP file on Local virtual machine using Xampp

1. Create a .php file in some local repository



2. Make changes in that .php file as per your desire and save it

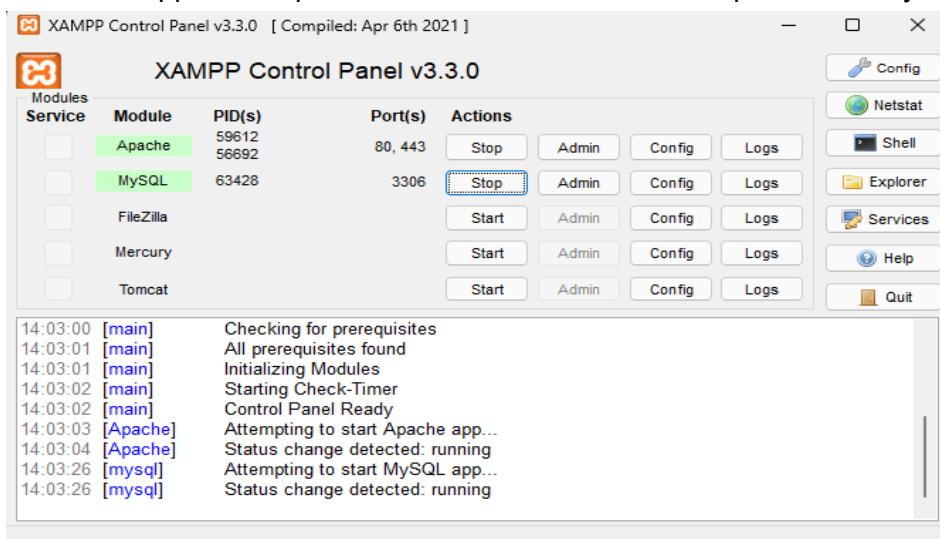
```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

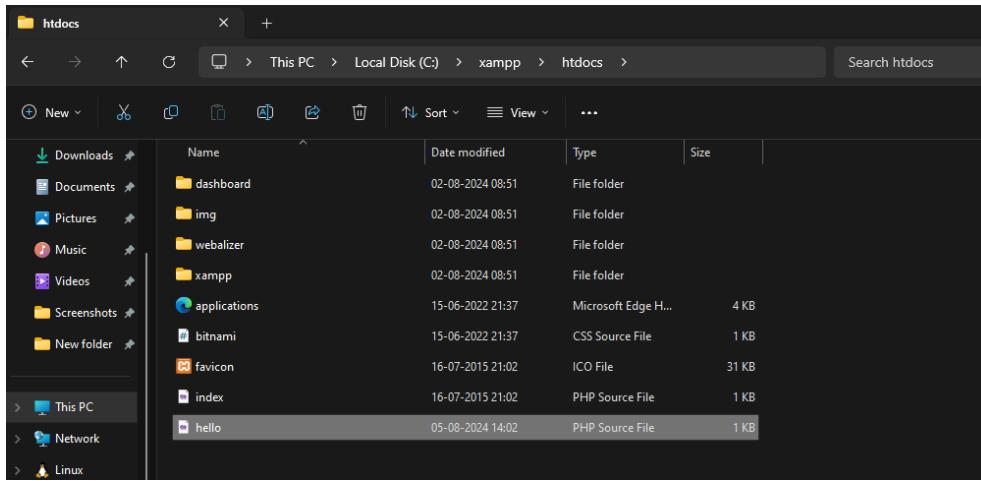
<?php
echo "Hello World!";
?>

</body>
</html>
```

3. Make sure you have installed Xampp on your local machine. After the installation, start Xampp control panel and start modules named Apache and MySql



4. Ensure that you relocate your .php file in the htdocs folder inside the Xampp folder



5. Access the contents of the php file by typing localhost/your_file.php on your browser. We have successfully hosted our php file on our local machine using Xampp.



b) Static hosting using AWS S3 bucket

1. Navigate to S3 inside services and create a bucket inside S3. Select the additional settings related to the bucket.

Buckets are containers for data stored in S3.

General configuration

AWS Region: US East (N. Virginia) us-east-1

Bucket type: General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

Directory - New
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name: myawsbucket05122004

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.

Choose bucket

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

ACLs disabled (recommended)
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

ACLs enabled
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership
Bucket owner enforced

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

2. Our bucket was successfully created. Now, we would want to add/upload our local files onto our bucket

Successfully created bucket "myawsbucket05122004". To upload files and folders, or to configure additional bucket settings, choose [View details](#).

Account snapshot - updated every 24 hours All AWS Regions
Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

[View Storage Lens dashboard](#)

General purpose buckets (1) [Info](#) All AWS Regions

Buckets are containers for data stored in S3.

Name	AWS Region	IAM Access Analyzer	Creation date
myawsbucket05122004	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 4, 2024, 22:40:10 (UTC+05:30)

myawsbucket05122004 [Info](#)

[Objects](#) [Properties](#) [Permissions](#) [Metrics](#) [Management](#) [Access Points](#)

Objects (0) [Info](#)

[Upload](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

[Find objects by prefix](#)

Name	Type	Last modified	Size	Storage class
No objects				

You don't have any objects in this bucket.

[Services](#) [Search](#) [Alt+S] N. Virginia vocabs/user3385491=Mohit_Kerkar

Upload [Info](#)

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose Add files or Add folder.

Files and folders (0)

All files and folders in this table will be uploaded.

[Find by name](#)

Name	Folder	Type
No files or folders		

You have not chosen any files or folders to upload.

3. Create a new html file, modify it and save it with a desired file name. Now, add/upload this file onto your S3 bucket.

Open [This PC](#) > Desktop > website1

Organize New folder

- website1
- OneDrive - Person
- This PC
- 3D Objects
- Desktop
- Documents
- Downloads
- Music
- Pictures
- Videos
- Windows (C:)

File name: hello Date modified: 04-08-2024 21:58

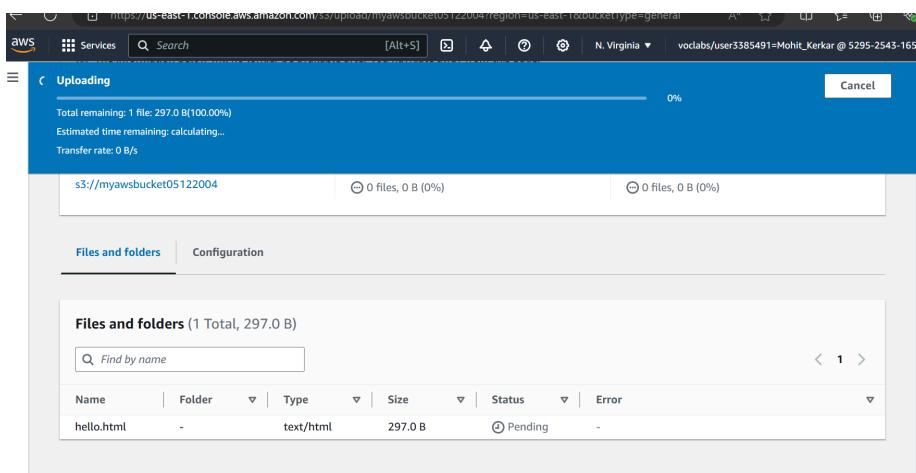
use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders here, or choose Add files or Add folder.

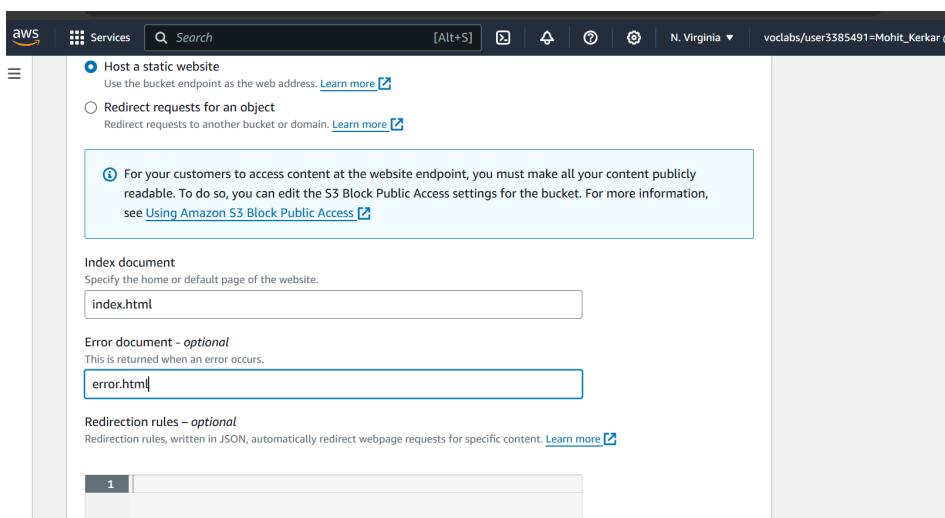
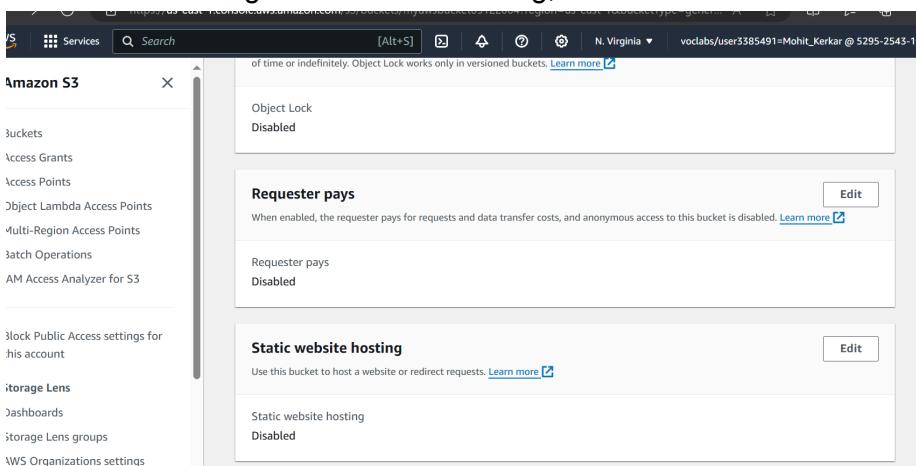
[Add files](#) [Add folder](#)

[Find by name](#)

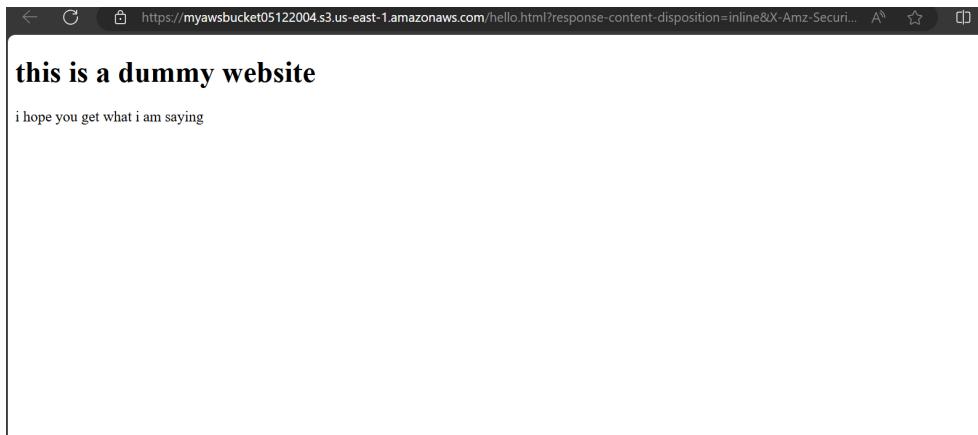
Name	Folder	Type
No files or folders		



4. For enabling static website hosting, we must enable this Static website hosting option



5. Open the website using the S3 bucket. You'd be able to see the contents of your html file.
Thus, we have successfully and statically hosted our html file using AWS S3 bucket



Conclusion: In conclusion, static hosting can be achieved locally using XAMPP or on AWS S3. With XAMPP, you can host PHP files on your local machine by placing them in the `htdocs` folder and accessing them via `localhost`. For cloud-based hosting, AWS S3 provides a simple solution by allowing you to upload HTML files to an S3 bucket and enable static website hosting. This approach requires minimal configuration and enables quick deployment of static content, making it ideal for small websites or testing environments.

Exp 02:To Build Your Application using AWS CodeBuild and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.

Step 1: Create our ElasticBeanstalk Environment

Login into your AWS account and navigate to services. Search for Elastic Beanstalk service and click on create application. Give your application a suitable name. For the platform, select PHP. Rest of the configuration settings are to be kept as default.

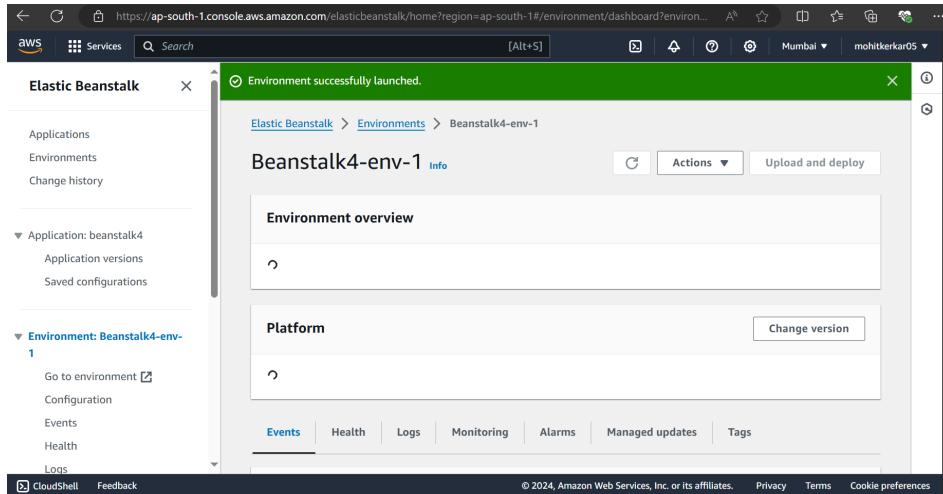
The screenshot shows the 'Create new application' form in the AWS Elastic Beanstalk console. The 'Application name' field contains 'beanstalk4'. The 'Tags' section is present but empty. The URL is https://ap-south-1.console.aws.amazon.com/elasticbeanstalk/home?region=ap-south-1#/create-application

The screenshot shows the 'Create environment' form in the AWS Elastic Beanstalk console. The 'Platform' dropdown is set to 'PHP'. The 'Platform branch' dropdown is set to 'PHP 8.3 running on 64bit Amazon Linux 2023'. The 'Platform version' dropdown is set to '4.3.1 (Recommended)'. The 'Application code' section has 'Sample application' selected. The URL is https://ap-south-1.console.aws.amazon.com/elasticbeanstalk/home?region=ap-south-1#/create-environment?applicationName=beanstalk4

Now, while creating the environment, we are asked to provide an IAM role with the necessary EC2 permissions. We are supposed to make sure that we have made an existing IAM role with the following set of permissions:

1. AWSElasticBeanstalkWebTier
2. AWSElasticBeanstalkWorkerTier
3. AWSElasticBeanstalkMulticontainerDocker

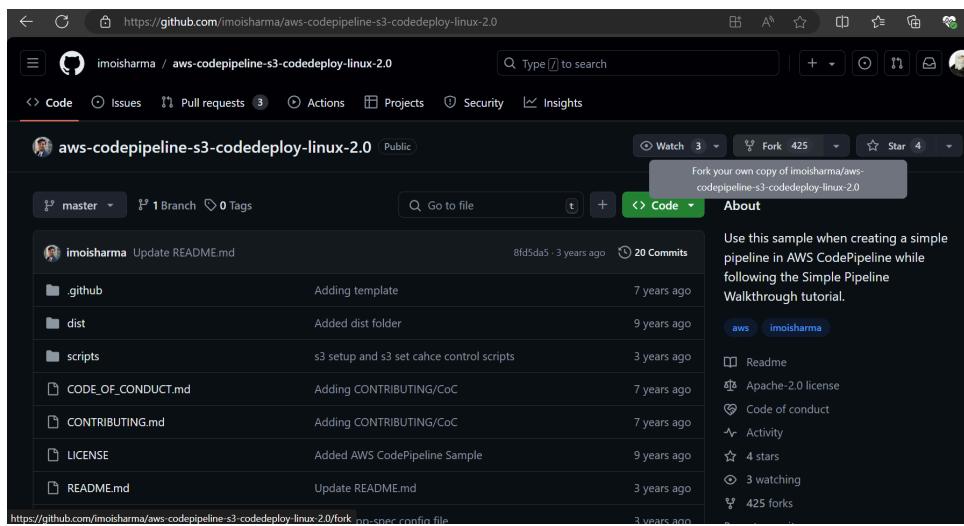
We can skip the steps to follow after the initial few steps mentioned above and move straight to review the settings of our environment. After reviewing everything properly, our environment can successfully be created.

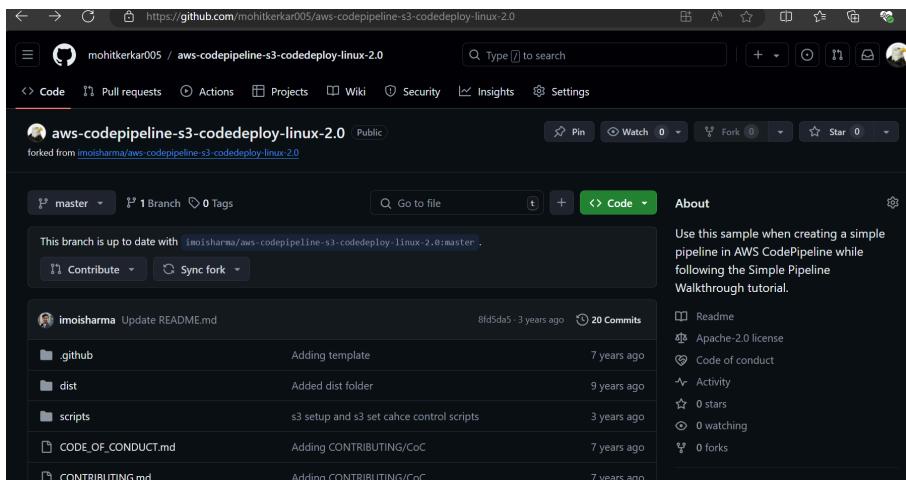


Step 2: Fork the required repository onto our github account

The repository to be forked is - imoisharma/aws-codepipeline-s3-codedeploy-linux-2.0

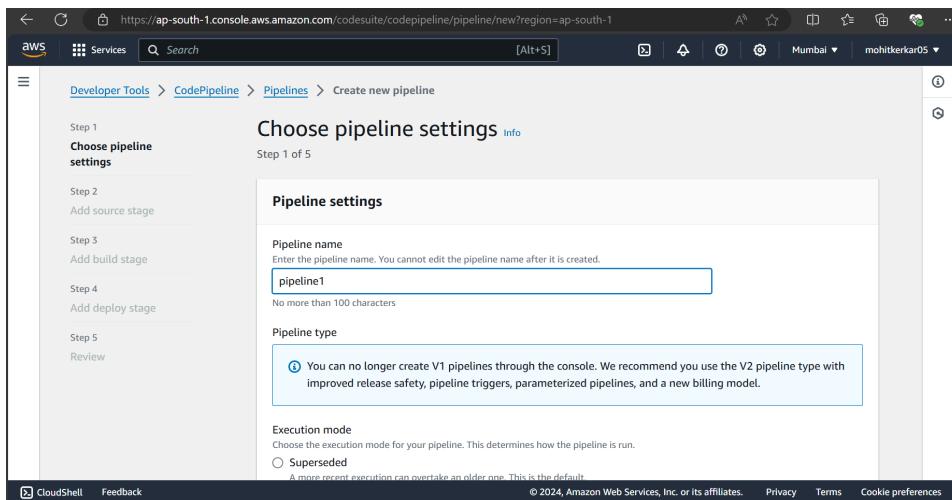
This step is necessary for the execution of the steps to follow. It will be helpful in the creation of a pipeline.



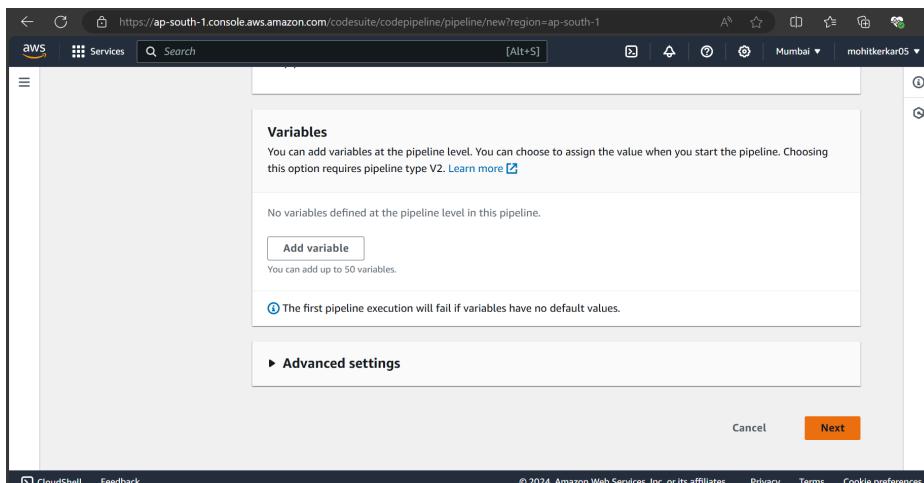


Step 3: Creation of the Pipeline

Navigate to Codepipeline inside Developer Tools. Give a suitable name to the pipeline you want to create.

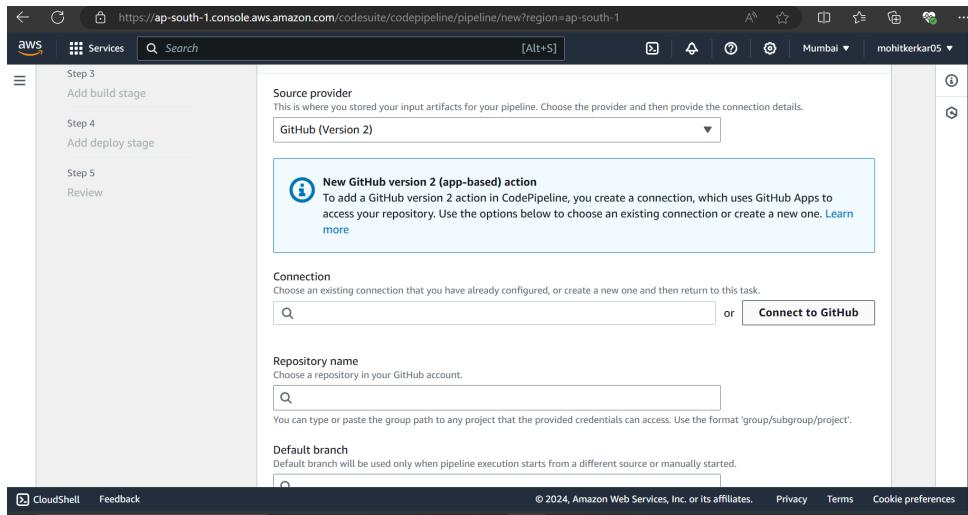


And click on next ...

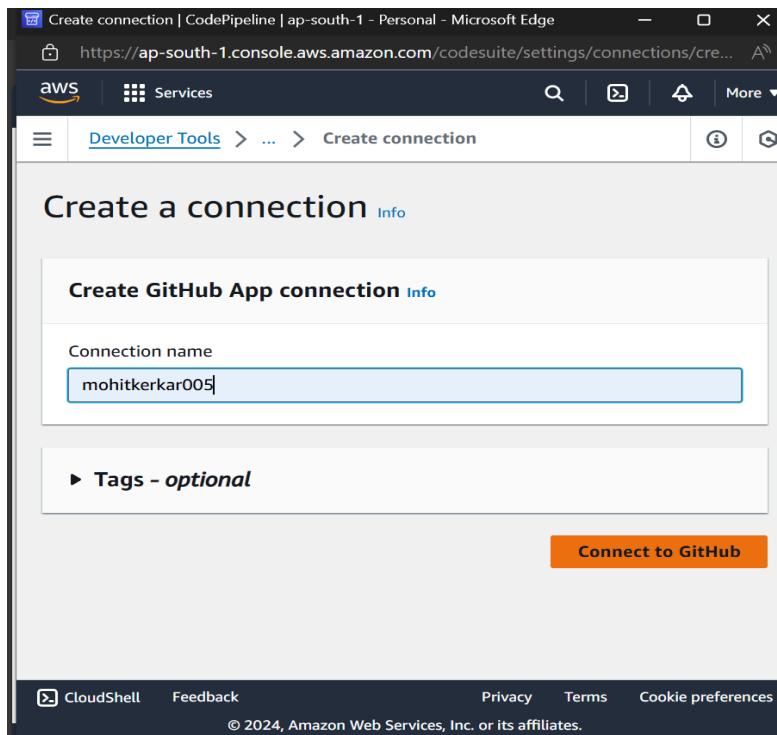


Step 4: Github connection

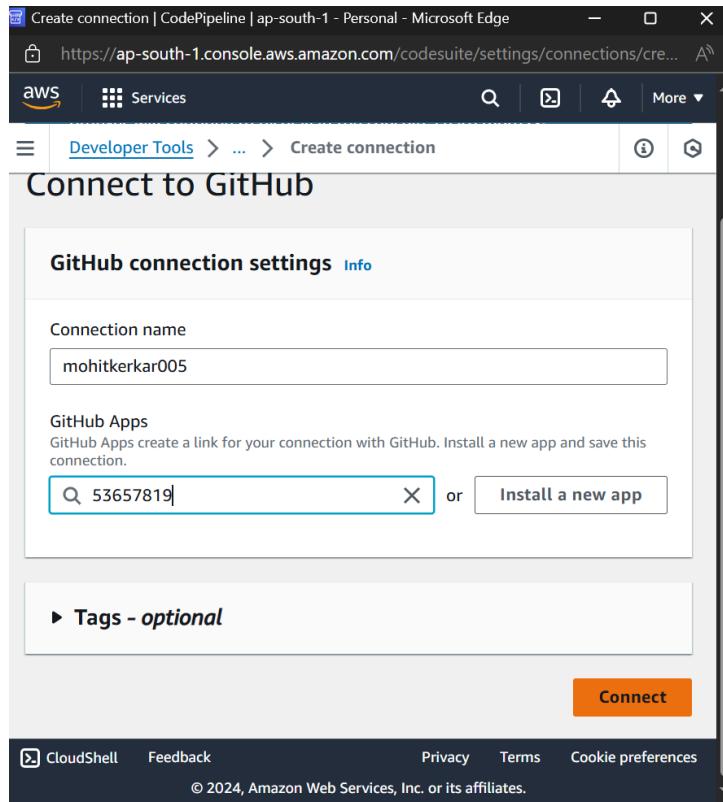
In this step, we are supposed to create a github connection and add our existing repository over here i.e the one we forked earlier



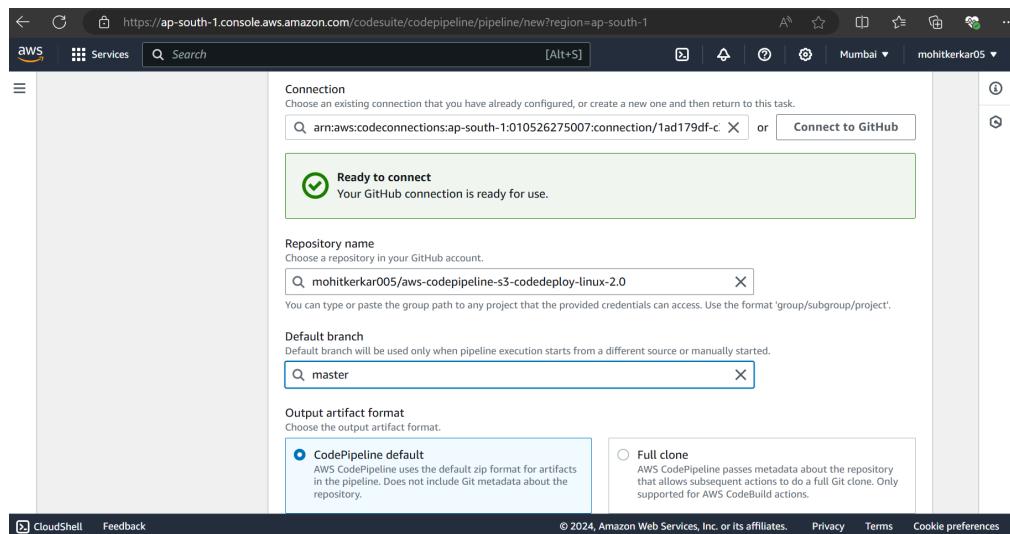
We are supposed to enter our github username so as to proceed towards making the connection



Now to finalize our connection, we are to install an application which connects AWS to our github account and repository.

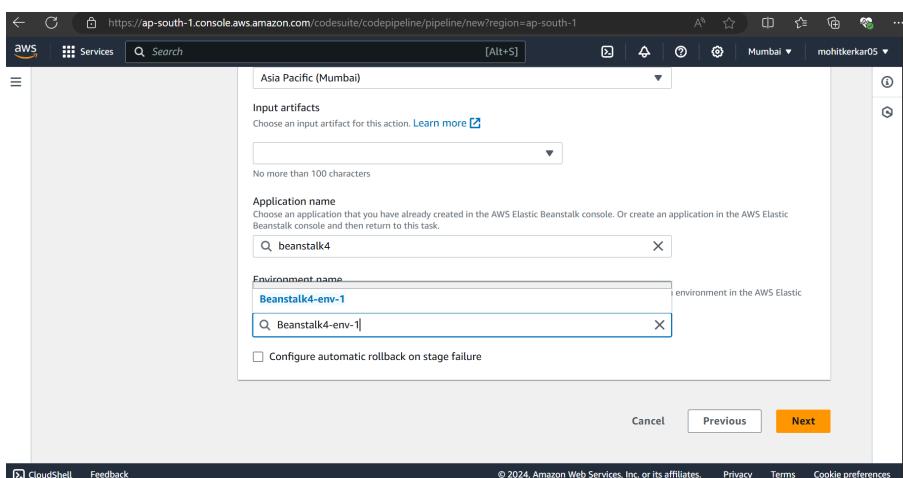
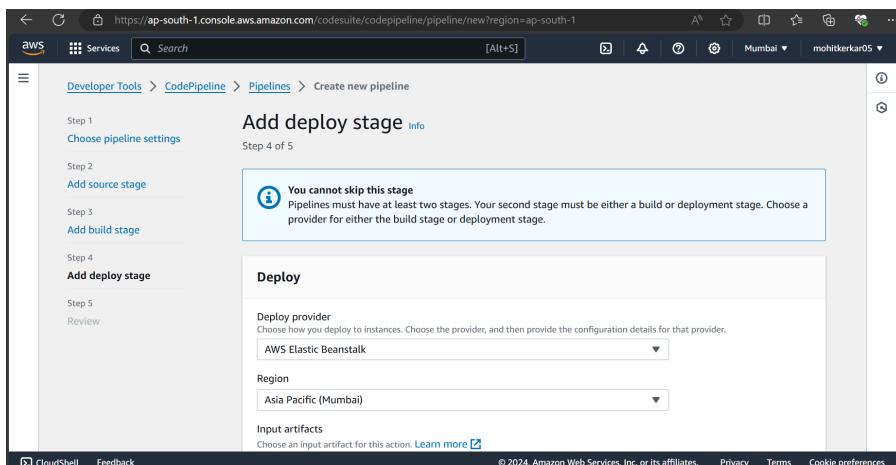


Post the establishment of the connection, this is the message that is displayed. We can further select the branch of our repository that we want to connect.

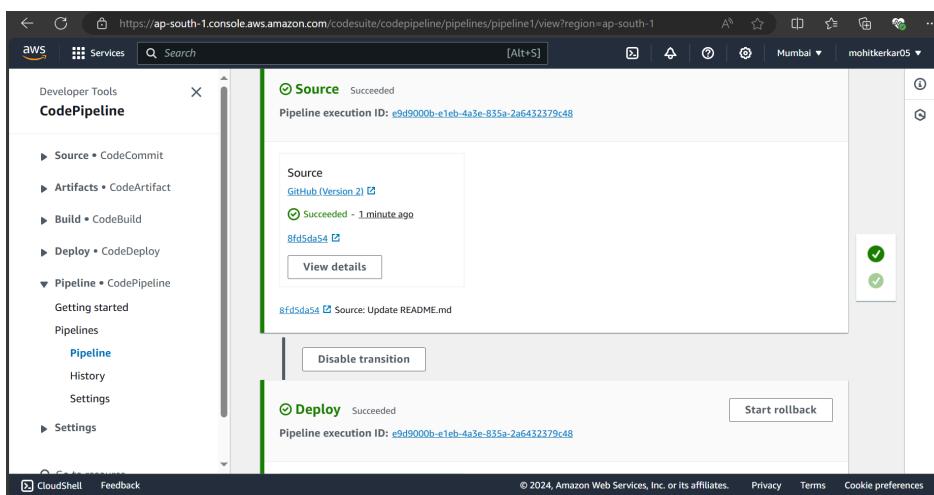


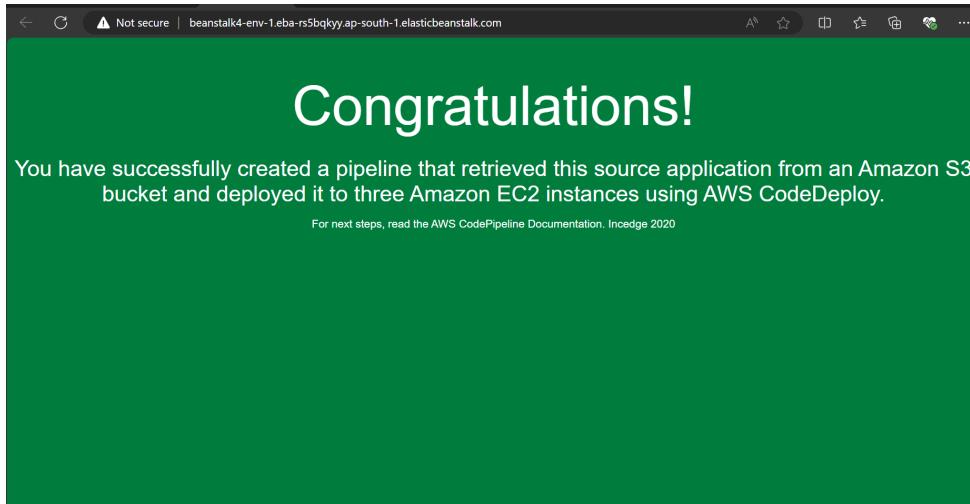
Step 5: Deployment stage:

We are expected to skip the build stage and move towards the deployment step. In the deployment step we are supposed to choose the Elastic Beanstalk application and the environment that we created earlier and proceed with our pipeline creation



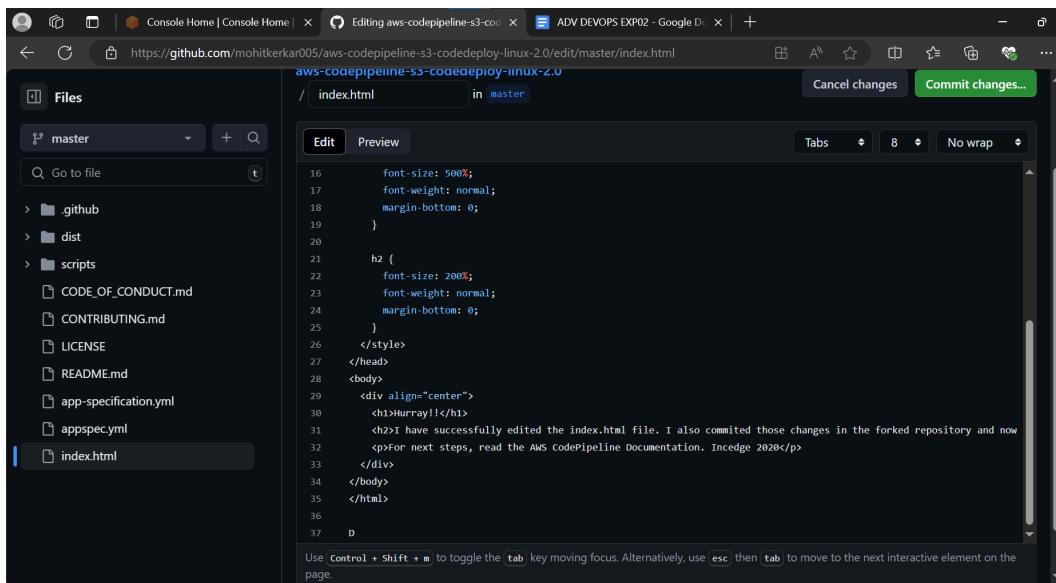
Step 6: Post deployment stage: When all the stages run successfully, this is what is displayed onto the screen. It shows us that our application and our environment have successfully been deployed using a dedicated pipeline created





Step 7: Committing changes to your github code

Now, we will go to our forked repository and make some changes to the index.html file. On making the desired changes, we are supposed to commit those changes on our forked repository. Write a good commit message so as to recognize it when it appears on the pipeline.



Step 8: Apply the newly made changes in index.html onto our pipeline

Come back to the Codepipeline section and select the pipeline through which we successfully created and deployed our application. Click on the release change option to apply the latest changes/commits from our github repository to our pipeline

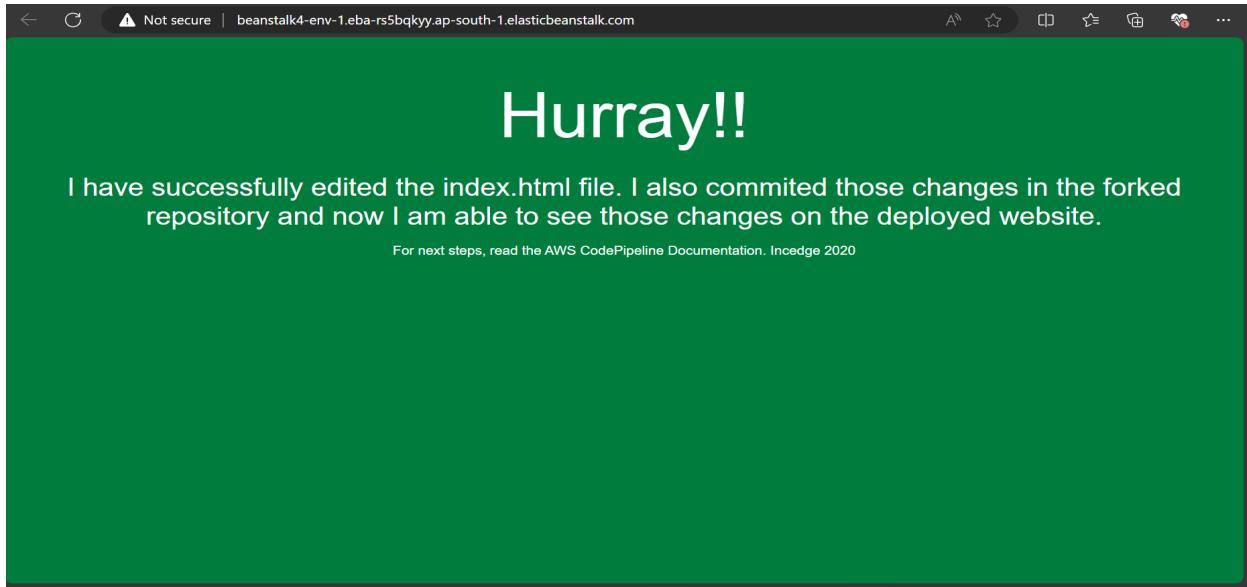
Name	Latest execution status	Latest source revisions	Latest execution started	Most recent executions
pipeline1 (Type: V2 Execution mode: QUEUED)	Succeeded	Source – 8fd5da54 Update README.md	17 hours ago	View details

Once the changes have been applied, we see the commit message that we wrote for the latest commit on our repository being reflected on our pipeline. Over here, it would be seen somewhere near the bottom of the image that is attached. “Update index.html” was the latest commit message in the github repository

Step 9: Open the Domain of our Elastic Beanstalk environment

Now, we navigate back to our Elastic Beanstalk environment and open the environment domain of our deployed application

The text in this image is clearly distinguishable from the earlier website’s text meaning that the changes that we made to our code in index.html has successfully been applied to the website that we deployed



Conclusion: In conclusion, this experiment demonstrates how to build and deploy an application using AWS CodeBuild, CodePipeline, and CodeDeploy. We created an Elastic Beanstalk environment, configured it with the required IAM roles, and connected it to a GitHub repository through CodePipeline. By making changes to the application code in the GitHub repository, we were able to update the deployed application seamlessly. This process highlights the effectiveness of AWS services in automating deployment, streamlining code integration, and ensuring that updates are reflected quickly in production environments.

Adv DevOps Exp 03

Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud

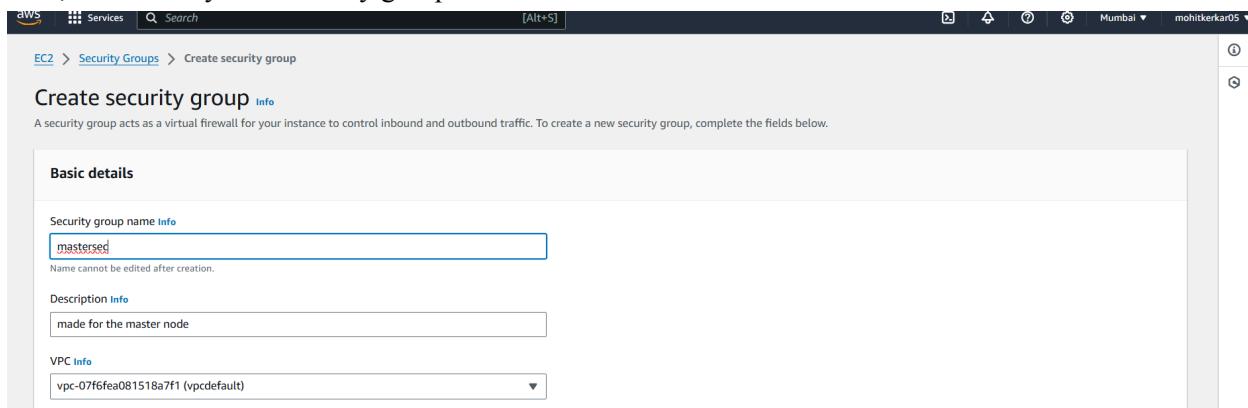
(I have performed this experiment on my personal AWS account)

Step 1: Create Key-pair, Security groups and required default VPCs

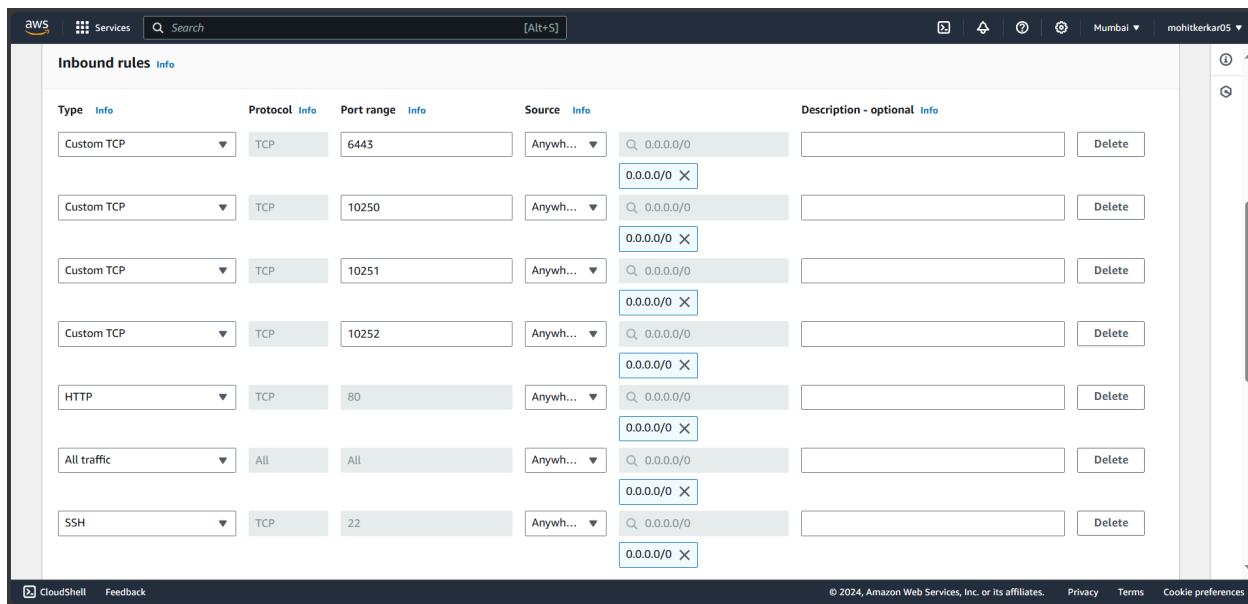
I started off by creating two separate security groups i.e one for the master instance (kubeadm to be initialized within it) and the other for the 2 worker instances. For creating any EC2 security groups, we require a VPC (Virtual private cloud) and a subnet along with it .. so that we can work with or allow inbound and outbound traffic and communication. Make sure you have a default VPC and a subnet already created which can be used for this experiment.

Also, make sure that you have a key pair installed of the type RSA with .pem extension on your local machine. Save it at a place which is accessible and where you can work from your terminal.

Here, i created my first security group



I modified the inbound rules in the following fashion for the master node



Then, i created a security group for the worker nodes

Edited the inbound rules for the worker nodes in the following fashion

Step 2: Create 3 EC2 instances i.e one master and other 2 workers

Now, in order to work with the instances, navigate to EC2 instances section and launch a few instances. Launch one instance and name it as masternode and the other two as worker.

I selected **AMI as ubuntu** among the list of OS that we shown available in the free tier eligibility list

Following is the master instance node:

Following is the worker instance node:

The screenshot shows the AWS EC2 'Create New Instance' wizard. In the 'Summary' step, the instance name is 'worker', the number of instances is 2, and the software image (AMI) is Canonical, Ubuntu, 24.04, amd64. The virtual server type is t3.micro. A tooltip indicates a free tier for the first year. The 'Launch instance' button is visible at the bottom.

IMPORTANT: Select t2.medium as the instance type for all 3 instances, as kubernetes requires at least 2 CPUs to work with and sufficient amount of other resources as well

The top screenshot shows the 'Instance type' configuration. The selected instance type is t2.medium, which has 2 vCPUs and 4 GiB Memory. The bottom screenshot shows the 'Network settings' configuration, where a security group named 'mastersec' is selected from a dropdown menu.

After launching all the 3 instances, select one instance and press ‘Connect’

The screenshot shows the AWS EC2 Instances page. It lists three instances under the 'Instances' section. The instances are:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
masternode	i-0264ba2470966fc5	Running	t3.medium	Initializing	View alarms +	ap-south-1c	-
worker	i-0f204d5780b6b582f	Running	t3.medium	Initializing	View alarms +	ap-south-1c	-
worker	i-0d551cff996f46f4f	Running	t3.medium	Initializing	View alarms +	ap-south-1c	-

Navigate to SSH client section and copy the command that's listed for connecting to the node remotely
`ssh -i "exp3key.pem" ubuntu@ec2-13-233-93-42.ap-south-1.compute.amazonaws.com`...for the master node
 Similarly, do this for all nodes

The screenshot shows the 'Connect to instance' page for the masternode instance (i-07f6d5a1bed9f4bf2). The 'SSH client' tab is selected. The page provides instructions for connecting:

- Open an SSH client.
- Locate your private key file. The key used to launch this instance is `exp3key.pem`.
- Run this command, if necessary, to ensure your key is not publicly viewable.
`chmod 400 "exp3key.pem"`
- Connect to your instance using its Public DNS:
`ec2-13-233-93-42.ap-south-1.compute.amazonaws.com`

Example:
`ssh -i "exp3key.pem" ubuntu@ec2-13-233-93-42.ap-south-1.compute.amazonaws.com`

Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Now, open 3 separate terminals and run change directories to the folder which contains the key we created earlier

Run their respective SSH commands (one in each terminal)

This helps us log onto those instances individually and remotely and work on them separately

```
ubuntu@ip-172-31-37-198: ~

PS C:\Users\DELL> cd C:\Users\DELL\Desktop\keypair
PS C:\Users\DELL\Desktop\keypair> ssh -i "exp3key.pem" ubuntu@ec2-3-111-188-84.ap-south-1.compute.amazonaws.com

The authenticity of host 'ec2-3-111-188-84.ap-south-1.compute.amazonaws.com (3.111.188.84)' can't be established.
ECDSA key fingerprint is SHA256:Lm8ajmtdu/3LBU1qu0mUjWlCHZLqFwhD1jaUCAQAOqM.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-111-188-84.ap-south-1.compute.amazonaws.com,3.111.188.84' (ECDSA) to the list
of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Thu Sep 26 09:43:59 UTC 2024

System load: 0.34          Processes:           118
Usage of /: 22.8% of 6.71GB  Users logged in:      0
Memory usage: 6%            IPv4 address for enx0: 172.31.37.198
Swap usage:  0%             

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.
```

```
PS C:\Windows\system32> cd C:\Users\DELL\Desktop\keypair
PS C:\Users\DELL\Desktop\keypair> ssh -i "exp3key.pem" ubuntu@ec2-13-234-38-84.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-13-234-38-84.ap-south-1.compute.amazonaws.com (13.234.38.84)' can't be established.
ECDSA key fingerprint is SHA256:P0+5r+bZ60Gpc0sL2A0+kkZYBNHMjfugne/ZBHMPHQ.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-13-234-38-84.ap-south-1.compute.amazonaws.com,13.234.38.84' (ECDSA) to the list
of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Thu Sep 26 09:46:24 UTC 2024

System load: 0.08      Processes:           118
Usage of /: 22.8% of 6.71GB  Users logged in:    0
Memory usage: 5%          IPv4 address for enX0: 172.31.47.161
Swap usage:  0%          
```

```
PS C:\Windows\system32> cd C:\Users\DELL\Desktop\keypair
PS C:\Users\DELL\Desktop\keypair> ssh -i "exp3key.pem" ubuntu@ec2-65-0-74-51.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-65-0-74-51.ap-south-1.compute.amazonaws.com (65.0.74.51)' can't be established.
ECDSA key fingerprint is SHA256:RTmw0J12RUM5s3vV2bYLL681AJE4iEasupz4bVTCjFw.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-65-0-74-51.ap-south-1.compute.amazonaws.com,65.0.74.51' (ECDSA) to the list of
known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Thu Sep 26 09:47:05 UTC 2024

System load: 0.13      Processes:           119
Usage of /: 22.8% of 6.71GB  Users logged in:    0
Memory usage: 5%          IPv4 address for enX0: 172.31.33.106
Swap usage:  0%          
```

Expanded Security Maintenance for Applications is not enabled.

Step 3: Docker installation:

Run the following command: These commands are used to install Docker on an Ubuntu system by adding Docker's official GPG key and configuring the Docker repository.

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee
/etc/apt/trusted.gpg.d/docker.gpg > /dev/null
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"
```

```
ubuntu@ip-172-31-37-198: ~
ubuntu@ip-172-31-37-198: $ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null
ubuntu@ip-172-31-37-198: $ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
> $(lsb_release -cs) stable"
Repository: deb [arch=amd64] https://download.docker.com/linux/ubuntu noble stable
Description:
Archive for codename: noble components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository...
Press [ENTER] to continue or Ctrl-c to cancel...
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:7 https://download.docker.com/linux/ubuntu/noble/universe Translation-en [5982 kB]
Get:8 https://download.docker.com/linux/ubuntu/noble/stable amd64 Packages [15.3 kB]
Get:9 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:10 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu/noble/universe amd64 c-n-f Metadata [301 kB]
Get:11 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu/multiverse amd64 Packages [269 kB]
Get:12 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu/noble/multiverse Translation-en [118 kB]
Get:13 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu/noble/multiverse amd64 Components [35.0 kB]
Get:14 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu/noble/multiverse amd64 c-n-f Metadata [8328 B]
```

Run the following commands to refresh your local package list to ensure the latest packages are available and install Docker Community Edition on your system without prompting for confirmation.

```
sudo apt-get update
```

```
sudo apt-get install -y docker-ce
```

```
ubuntu@ip-172-31-37-198: ~
ubuntu@ip-172-31-37-198: $ sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignored as the file has an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
ubuntu@ip-172-31-37-198: $ sudo apt-get install -y docker-ce
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7
  libslirp0 pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin
  libltdl7 libslirp0 pigz slirp4netns
0 upgraded, 10 newly installed, 0 to remove and 142 not upgraded.
Need to get 123 MB of archives.
After this operation, 442 MB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
```

```
ubuntu@ip-172-31-37-198: ~
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /usr/lib/systemd/system/containerd.service.
Setting up docker-compose-plugin (2.29.7-1~ubuntu.24.04~noble) ...
Setting up libltdl7:amd64 (2.4.7-7build1) ...
Setting up docker-ce-cli (5:27.3.1-1~ubuntu.24.04~noble) ...
Setting up libslirp0:amd64 (4.7.0-1ubuntu3) ...
Setting up pigz (2.8-1) ...
Setting up docker-ce-rootless-extras (5:27.3.1-1~ubuntu.24.04~noble) ...
Setting up slirp4netns (1.2.1-1build2) ...
Setting up docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

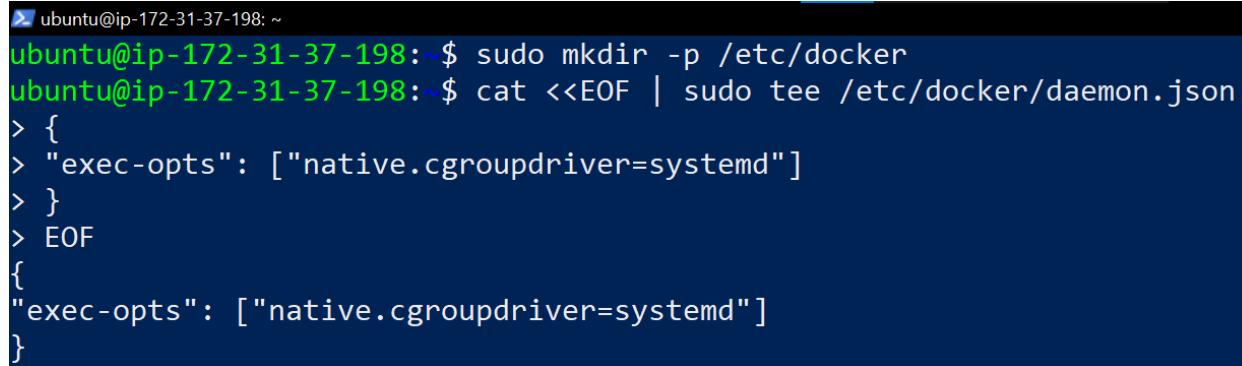
No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
```

Run the following commands to create the Docker configuration directory and write a configuration file for Docker to use the systemd cgroup driver.

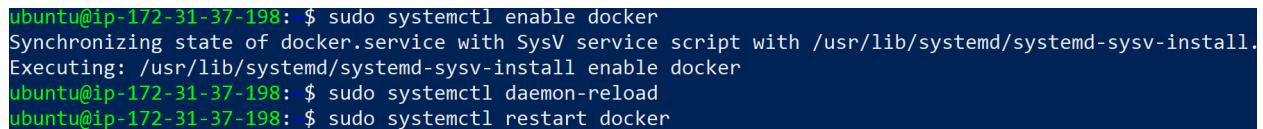
```
sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
```



```
ubuntu@ip-172-31-37-198:~$ sudo mkdir -p /etc/docker
ubuntu@ip-172-31-37-198:~$ cat <<EOF | sudo tee /etc/docker/daemon.json
> {
>   "exec-opts": ["native.cgroupdriver=systemd"]
> }
> EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}
```

Run the following commands to ensure Docker starts automatically on system boot, reload systemd to apply new configurations, restart Docker to apply the new configuration.

```
sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
```

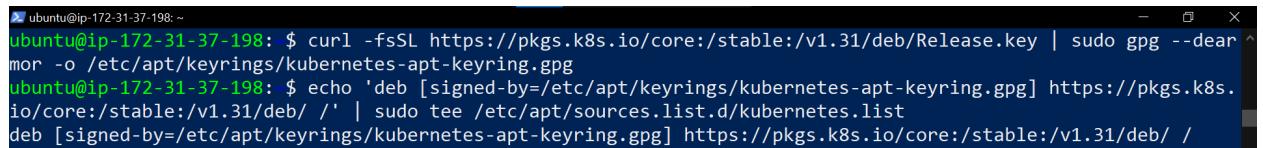


```
ubuntu@ip-172-31-37-198:~$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@ip-172-31-37-198:~$ sudo systemctl daemon-reload
ubuntu@ip-172-31-37-198:~$ sudo systemctl restart docker
```

Step 4: Kubernetes Installation:

Run the following commands to add the Kubernetes package repository to your Ubuntu system in order to install Kubernetes components like kubectl, kubelet, and kubeadm.

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/' | sudo tee /etc/apt/sources.list.d/kubernetes.list
```



```
ubuntu@ip-172-31-37-198:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
ubuntu@ip-172-31-37-198:~$ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/' | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

Next, run these commands to refresh the local package index to include the newly added Kubernetes repository, install the main Kubernetes components (kubelet, kubeadm, kubectl), prevent the installed Kubernetes components from being automatically updated, ensuring cluster stability until manual updates are performed.

```
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

```
ubuntu@ip-172-31-37-198:~$ sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [533 kB]
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [129 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [376 kB]
Hit:8 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:9 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [155 kB]
Get:10 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb InRelease [1186 B]
Get:11 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb Packages [4865 B]
Fetched 1324 kB in 1s (2118 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignored as the file has an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
ubuntu@ip-172-31-37-198:~$ sudo apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  kubernetes-cni
Preparing to unpack .../3-kubectl_1.31.1-1.1_amd64.deb ...
Unpacking kubectl (1.31.1-1.1) ...
Selecting previously unselected package kubernetes-cni.
Preparing to unpack .../4-kubernetes-cni_1.5.1-1.1_amd64.deb ...
Unpacking kubernetes-cni (1.5.1-1.1) ...
Selecting previously unselected package kubelet.
Preparing to unpack .../5-kubelet_1.31.1-1.1_amd64.deb ...
Unpacking kubelet (1.31.1-1.1) ...
Setting up conntrack (1:1.4.8-1ubuntu1) ...
Setting up kubectl (1.31.1-1.1) ...
Setting up cri-tools (1.31.1-1.1) ...
Setting up kubernetes-cni (1.5.1-1.1) ...
Setting up kubeadm (1.31.1-1.1) ...
Setting up kubelet (1.31.1-1.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-37-198:~$ sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
```

The command **sudo systemctl enable --now kubelet** enables the **kubelet** service, ensuring it starts automatically at boot and immediately starts running without needing a system reboot. The **sudo apt-get install -y containerd** command installs **containerd**, a container runtime that Kubernetes uses to manage and run containers.

```
sudo systemctl enable --now kubelet
sudo apt-get install -y containerd
```

```
ubuntu@ip-172-31-37-198: ~
ubuntu@ip-172-31-37-198: $ sudo systemctl enable --now kubelet
ubuntu@ip-172-31-37-198: $ sudo apt-get install -y containerd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz
  slirp4netns
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  runc
The following packages will be REMOVED:
  containerd.io docker-ce
The following NEW packages will be installed:
  containerd runc
0 upgraded, 2 newly installed, 2 to remove and 142 not upgraded.
Need to get 47.2 MB of archives.
After this operation, 53.1 MB disk space will be freed.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.1.12-0ubuntu3.1 [85
99 kB]
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.12-0ubuntu4
.1 [38.6 MB]
Fetched 47.2 MB in 1s (54.9 MB/s)
(Reading database ... 68064 files and directories currently installed.)
Removing docker-ce (5:27.3.1-1~ubuntu.24.04-noble) ...
Removing containerd.io (1.7.22-1) ...
Selecting previously unselected package runc.
```

The command `sudo mkdir -p /etc/containerd` creates the directory for containerd configuration files if it doesn't already exist. The next command, `sudo containerd config default | sudo tee /etc/containerd/config.toml`, generates a default configuration for containerd and saves it as `config.toml` in that directory. Together, these commands set up the necessary directory and create a default configuration file for containerd.

```
sudo mkdir -p /etc/containerd
```

```
sudo containerd config default | sudo tee /etc/containerd/config.toml
```

```
ubuntu@ip-172-31-37-198: ~
ubuntu@ip-172-31-37-198: $ sudo mkdir -p /etc/containerd
ubuntu@ip-172-31-37-198: $ sudo containerd config default | sudo tee /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
  path = ""

[debug]
  address = ""
  format = ""
  gid = 0
  level = ""
  uid = 0

[grpc]
  address = "/run/containerd/containerd.sock"
  gid = 0
  max_recv_message_size = 16777216
  max_send_message_size = 16777216
```

The command `sudo systemctl restart containerd` restarts the `containerd` service to apply any changes. `sudo systemctl enable containerd` sets it to start automatically at boot, while `sudo systemctl status containerd` checks its current status, showing whether it's active and any errors. Together, these commands manage the operation and health of the container runtime.

```
sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd
```

```
ubuntu@ip-172-31-37-198:~$ sudo systemctl restart containerd
ubuntu@ip-172-31-37-198:~$ sudo systemctl enable containerd
ubuntu@ip-172-31-37-198:~$ sudo systemctl status containerd
● containerd.service - containerd container runtime
   Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Thu 2024-09-26 10:09:37 UTC; 14s ago
       Docs: https://containerd.io
      Main PID: 4771 (containerd)
        Tasks: 7
         Memory: 13.0M (peak: 13.4M)
            CPU: 123ms
           CGroup: /system.slice/containerd.service
                     └─4771 /usr/bin/containerd

Sep 26 10:09:37 ip-172-31-37-198 containerd[4771]: time="2024-09-26T10:09:37.127045263Z" level=info msg="Starting containerd"
Sep 26 10:09:37 ip-172-31-37-198 containerd[4771]: time="2024-09-26T10:09:37.127119958Z" level=info msg="Starting containerd"
Sep 26 10:09:37 ip-172-31-37-198 containerd[4771]: time="2024-09-26T10:09:37.127202179Z" level=info msg="Starting containerd"
Sep 26 10:09:37 ip-172-31-37-198 containerd[4771]: time="2024-09-26T10:09:37.127214694Z" level=info msg="Starting containerd"
Sep 26 10:09:37 ip-172-31-37-198 containerd[4771]: time="2024-09-26T10:09:37.127239102Z" level=info msg="Starting containerd"
Sep 26 10:09:37 ip-172-31-37-198 containerd[4771]: time="2024-09-26T10:09:37.127260747Z" level=info msg="Starting containerd"
Sep 26 10:09:37 ip-172-31-37-198 containerd[4771]: time="2024-09-26T10:09:37.127807399Z" level=info msg="servicing"
Sep 26 10:09:37 ip-172-31-37-198 containerd[4771]: time="2024-09-26T10:09:37.127910878Z" level=info msg="servicing"
Sep 26 10:09:37 ip-172-31-37-198 containerd[4771]: time="2024-09-26T10:09:37.128176100Z" level=info msg="containing"
Sep 26 10:09:37 ip-172-31-37-198 systemd[1]: Started containerd.service - containerd container runtime.
```

Socat installation:

```
sudo apt-get install -y socat
```

```
ubuntu@ip-172-31-37-198:~$ sudo apt-get install -y socat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz
  slirp4netns
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  socat
0 upgraded, 1 newly installed, 0 to remove and 142 not upgraded.
Need to get 374 kB of archives.
After this operation, 1649 kB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble amd64 socat amd64 1.8.0.0-4build3 [374 kB]
Fetched 374 kB in 0s (16.7 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68108 files and directories currently installed.)
Preparing to unpack .../socat_1.8.0.0-4build3_amd64.deb ...
Unpacking socat (1.8.0.0-4build3) ...
Setting up socat (1.8.0.0-4build3) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.
```

Step 5: Kubernetes Cluster

Run this command only on the master instance

```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
```

It initializes a Kubernetes cluster with kubeadm and sets up the control plane (master node).

```
ubuntu@ip-172-31-37-198: ~
ubuntu@ip-172-31-37-198: $ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'
W0926 10:13:04.272663      5083 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the
container runtime is inconsistent with that used by kubeadm. It is recommended to use "registry.k8s.io/pause:3.
10" as the CRI sandbox image.
[certs] Using certificatebir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [ip-172-31-37-198 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.37.198]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [ip-172-31-37-198 localhost] and IPs [172.31.37.198 12
7.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-37-198 localhost] and IPs [172.31.37.198 127.
0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
```

```
luster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate a
nd key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy
```

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Alternatively, if you are the root user, you can run:

```
export KUBECONFIG=/etc/kubernetes/admin.conf
```

You should now deploy a pod network to the cluster.

Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
<https://kubernetes.io/docs/concepts/cluster-administration/addons/>

Then you can join any number of worker nodes by running the following on each as root:

```
kubeadm join 172.31.37.198:6443 --token gci45u.g7qowjugiqw1ynrb \
--discovery-token-ca-cert-hash sha256:a984e32c6c7c4815973519ebb45d64bf44a23f4e48dbf1195d8f0e695ea9409e
```

Run this command on master and also copy and save the Join command from above.

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

```
ubuntu@ip-172-31-37-198:~$ mkdir -p $HOME/.kube
ubuntu@ip-172-31-37-198:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
ubuntu@ip-172-31-37-198:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-37-198:~$ kubectl get nodes
NAME           STATUS      ROLES      AGE      VERSION
ip-172-31-37-198   NotReady   control-plane   12m     v1.31.1
```

Connect master and worker nodes by running this command on the worker logged in terminals:

I ran the following command to do this,

```
kubeadm join 172.31.37.198:6443 --token gci45u.g7qowjugiqw1ynrb \ --discovery-token-ca-cert-hash
sha256:a984e32c6c7c4815973519ebb45d64bf44a23f4e48dbf1195d8f0e695ea9409e
```

On node 1:

```
ubuntu@ip-172-31-47-161:~$ sudo kubeadm join 172.31.37.198:6443 --token gci45u.g7qowjugiqw1ynrb --discovery-token-ca-cert-hash sha256:a984e32c6c7c4815973519ebb45d64bf44a23f4e48dbf1195d8f0e695ea9409e
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.001901347s
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

ubuntu@ip-172-31-47-161:~$
```

On node 2:

```
ubuntu@ip-172-31-33-106:~$ sudo kubeadm join 172.31.37.198:6443 --token gci45u.g7qowjugiqw1ynrb --discovery-token-ca-cert-hash sha256:a984e32c6c7c4815973519ebb45d64bf44a23f4e48dbf1195d8f0e695ea9409e
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.002345051s
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Run kubectl get nodes on the master instance to confirm the joining of the worker nodes

```
Select ubuntu@ip-172-31-37-198:~
ubuntu@ip-172-31-37-198:~$ kubectl get nodes
NAME           STATUS      ROLES      AGE      VERSION
ip-172-31-33-106   NotReady   <none>      17s     v1.31.1
ip-172-31-37-198   NotReady   control-plane   18m     v1.31.1
ip-172-31-47-161   NotReady   <none>      39s     v1.31.1
```

Since Status is NotReady we have to add a network plugin. And also we have to give the name to the nodes.

kubectl apply -f <https://docs.projectcalico.org/manifests/calico.yaml>

The command kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml deploys Calico, a networking and network security solution for Kubernetes, by applying the configuration specified in the provided YAML file from the Calico documentation. This sets up the necessary resources to enable networking capabilities within the Kubernetes cluster.

```
Select ubuntu@ip-172-31-37-198: ~
ubuntu@ip-172-31-37-198: $ kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
podo disruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
configmap/calico-config created
customresourcedefinition.apixtensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apixtensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apixtensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apixtensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apixtensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apixtensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apixtensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apixtensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apixtensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apixtensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apixtensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apixtensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apixtensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apixtensions.k8s.io/ippreservations.crd.projectcalico.org created
customresourcedefinition.apixtensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apixtensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apixtensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
daemonset.apps/calico-node created
```

sudo systemctl status kubelet

```
Select ubuntu@ip-172-31-37-198: ~
ubuntu@ip-172-31-37-198: $ sudo systemctl status kubelet
● kubelet.service - kubelet: The Kubernetes Node Agent
  Loaded: loaded (/usr/lib/systemd/system/kubelet.service; enabled; preset: enabled)
  Drop-In: /usr/lib/systemd/system/kubelet.service.d
            └─10-kubeadm.conf
    Active: active (running) since Thu 2024-09-26 10:13:42 UTC; 18min ago
      Docs: https://kubernetes.io/docs/
   Main PID: 5773 (kubelet)
     Tasks: 10 (limit: 4676)
    Memory: 32.6M (peak: 33.3M)
       CPU: 18.442s
      CGroup: /system.slice/kubelet.service
              └─5773 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/ssl/kubelet.pem

Sep 26 10:32:24 ip-172-31-37-198 kubelet[5773]: I0926 10:32:24.596915 5773 reconciler_common.go:245] "oper>
Sep 26 10:32:24 ip-172-31-37-198 kubelet[5773]: I0926 10:32:24.596938 5773 reconciler_common.go:245] "oper>
Sep 26 10:32:24 ip-172-31-37-198 kubelet[5773]: I0926 10:32:24.596958 5773 reconciler_common.go:245] "oper>
Sep 26 10:32:24 ip-172-31-37-198 kubelet[5773]: I0926 10:32:24.596978 5773 reconciler_common.go:245] "oper>
Sep 26 10:32:24 ip-172-31-37-198 kubelet[5773]: I0926 10:32:24.596997 5773 reconciler_common.go:245] "oper>
Sep 26 10:32:24 ip-172-31-37-198 kubelet[5773]: I0926 10:32:24.597017 5773 reconciler_common.go:245] "oper>
Sep 26 10:32:25 ip-172-31-37-198 kubelet[5773]: I0926 10:32:25.193047 5773 scope.go:117] "RemoveContainer" >
Sep 26 10:32:25 ip-172-31-37-198 kubelet[5773]: E0926 10:32:25.193206 5773 pod_workers.go:1301] "Error sync>
Sep 26 10:32:27 ip-172-31-37-198 kubelet[5773]: E0926 10:32:27.544395 5773 kubelet.go:2902] "Container runt>
Sep 26 10:32:32 ip-172-31-37-198 kubelet[5773]: E0926 10:32:32.545352 5773 kubelet.go:2902] "Container runt>
lines 1-23/23 (END)
ubuntu@ip-172-31-37-198: $
```

kubectl get nodes -o wide helps us get to know that the Status is ready.

```
ubuntu@ip-172-31-37-198: $ kubectl get nodes -o wide
NAME           STATUS   ROLES      AGE    VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE   KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-33-106   Ready    <none>    66s   v1.31.1   172.31.33.106   <none>       Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-37-198   Ready    control-plane   19m   v1.31.1   172.31.37.198   <none>       Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-47-161   Ready    <none>    88s   v1.31.1   172.31.47.161   <none>       Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ubuntu@ip-172-31-37-198: $ kubectl label node ip-172-31-28-117 kubernetes.io/role=Node1
Error from server (NotFound): nodes "ip-172-31-28-117" not found
ubuntu@ip-172-31-37-198: $ kubectl label node ip-172-31-47-161 kubernetes.io/role=Node1
node/ip-172-31-47-161 labeled
ubuntu@ip-172-31-37-198: $ kubectl label node ip-172-31-33-106 kubernetes.io/role=Node2
node/ip-172-31-33-106 labeled
ubuntu@ip-172-31-37-198: $ kubectl get nodes -o wide
NAME           STATUS   ROLES      AGE    VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE   KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-33-106   Ready    Node2     3m26s  v1.31.1   172.31.33.106   <none>       Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-37-198   Ready    control-plane   21m   v1.31.1   172.31.37.198   <none>       Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-47-161   Ready    Node1     3m48s  v1.31.1   172.31.47.161   <none>       Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
```

Or else run kubectl get nodes command

```
ubuntu@ip-172-31-37-198:~$ kubectl get nodes
NAME           STATUS   ROLES      AGE    VERSION
ip-172-31-33-106   Ready    Node2     3m42s  v1.31.1
ip-172-31-37-198   Ready    control-plane   21m   v1.31.1
ip-172-31-47-161   Ready    Node1     4m4s   v1.31.1
ubuntu@ip-172-31-37-198:~$ ■
```

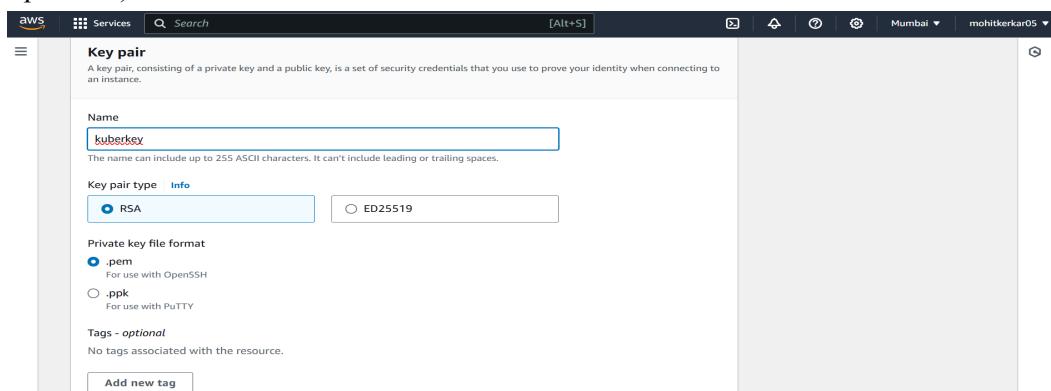
Conclusion: In this experiment, we successfully set up a Kubernetes cluster on AWS by creating the necessary infrastructure and configuring security groups. We installed Docker and Kubernetes components, initialized the master node, and connected the worker nodes. By deploying Calico for networking, we enabled effective communication within the cluster. This hands-on experience enhanced our understanding of Kubernetes architecture and equipped us with practical skills for managing containerized applications in a cloud environment.

Adv DevOps Exp 04

Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.
(I performed this Experiment on my personal AWS account)

Step 1: Creation of EC2 Instance, its Key-pair, remote login using SSH

Firstly, create a key pair with a desired name and type ‘RSA’ with .pem extension. (.pem is useful with OpenSSH)

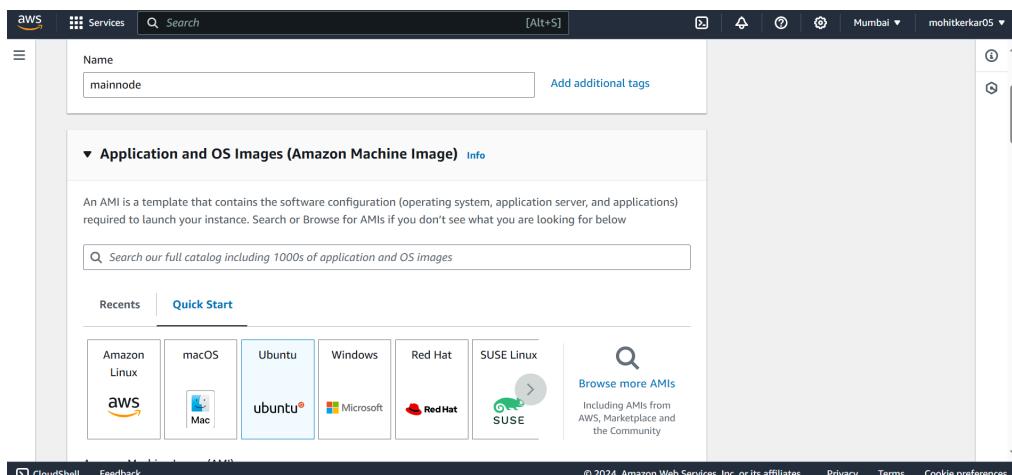


After creating the key, it automatically gets downloaded onto your machine. We are to create a new folder and place our within there. After this, we are to access our key and establish our connection with our instance by changing the directory to our key’s directory

Now, proceed to the creation of the EC2 instance. I selected Amazon machine image as **Ubuntu** and instance type as **t2.medium**, just for the purpose of convenience and also for successful execution of the experiment.

The execution of the experiment is determined by the fact that how many resources is the instance allowed to access or use. The Free Tier eligible instance type **t2.micro** **avails the instance with only one CPU**, whereas with t2.medium we get 2 CPU and much more execution space.

Be mindful that along with all this, **t2.medium does not have free tier eligibility**. So, use the instances and the resources that you allow them to access resourcefully and economically. Shut/terminate them as and when work related to them is over



▼ Instance type [Info](#) | [Get advice](#)

Instance type

t2.medium

Family: t2 2 vCPU 4 GiB Memory Current generation: true
 On-Demand Linux base pricing: 0.0496 USD per Hour
 On-Demand Windows base pricing: 0.0676 USD per Hour
 On-Demand RHEL base pricing: 0.0784 USD per Hour
 On-Demand SUSE base pricing: 0.1496 USD per Hour

Additional costs apply for AMIs with pre-installed software

Instances (1/1) [Info](#)

Last updated less than a minute ago

[Connect](#) [Actions](#) [Launch instances](#)

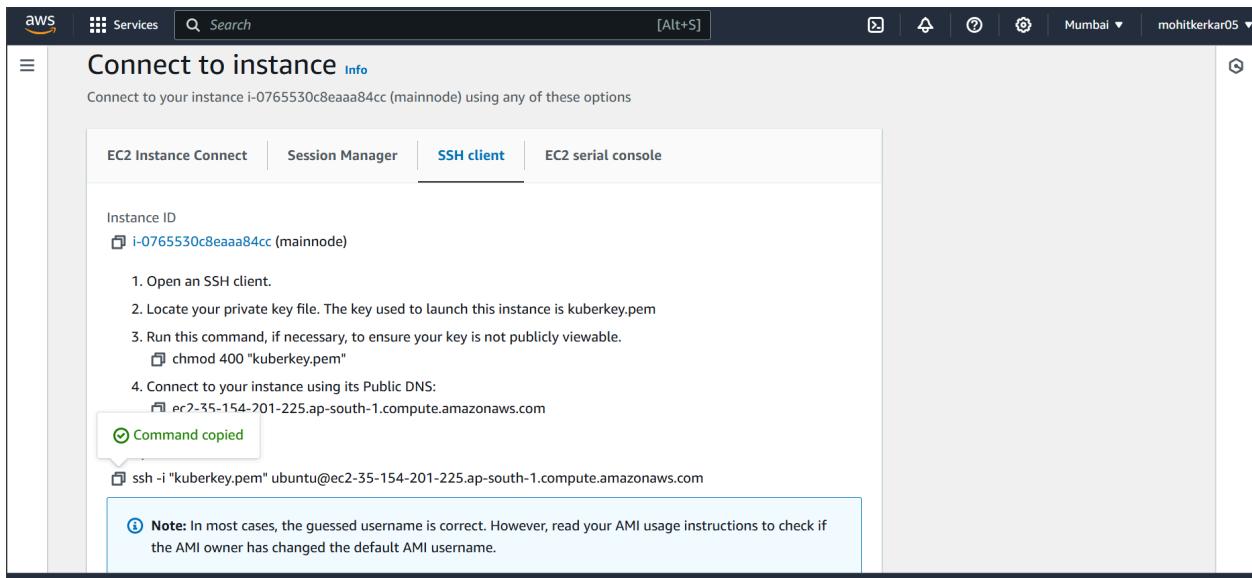
Name	Instance ID	Instance state	Instance type	Status check	Alarm status
mainnode	i-0765530c8aaaa84cc	Running	t2.medium	Initializing	View alarms +

Here are my instance details. From here on, after remotely logging in from my local terminal, you would see my IPv4 address in my terminal (Powershell)

i-0765530c8aaaa84cc (mainnode)

IPv6 address	35.154.201.225 open address	172.31.40.244
Hostname type	Instance state	Public IPv4 DNS
IP name: ip-172-31-40-244.ap-south-1.compute.internal	Running	ec2-35-154-201-225.ap-south-1.compute.amazonaws.com open address
	Private IP DNS name (IPv4 only)	

Select the instance and press connect. Navigate to the SSH client section and copy the SSH command provided to us. We will require this for remotely logging into our instance from our local terminal.



As one can see, I changed my directory to the folder in which i had stored my key (.pem) and ran the ssh command so as to remotely login onto my instance and perform the necessary tasks from my terminal itself

```
ubuntu@ip-172-31-40-244: ~
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> cd C:\Users\DEll\Desktop\keypair
PS C:\Users\DEll\Desktop\keypair> ssh -i "kuberkey.pem" ubuntu@ec2-35-154-201-225.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-35-154-201-225.ap-south-1.compute.amazonaws.com (35.154.201.225)' can't be established.
ECDSA key fingerprint is SHA256:GiS2bs4t0fWy44Hiy3aN3Li34BT8eDD+LxwJMADr7HU.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-35-154-201-225.ap-south-1.compute.amazonaws.com,35.154.201.225' (ECDSA) to the
list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Wed Sep 25 16:45:41 UTC 2024

System load:  0.0          Processes:           113
Usage of /:   22.8% of 6.71GB  Users logged in:     0
Memory usage: 5%
Swap usage:  0%
IPv4 address for enX0: 172.31.40.244

Expanded Security Maintenance for Applications is not enabled.
```

Step 2: Docker installation

Run the below mentioned commands in order to install docker on your instance

Adding key: curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

Adding key without apt-key: curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null

Add docker repository: sudo add-apt-repository "deb [arch=amd64]

https://download.docker.com/linux/ubuntu

\$(lsb_release -cs) stable"

```
ubuntu@ip-172-31-40-244: ~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
ubuntu@ip-172-31-40-244: ~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null
ubuntu@ip-172-31-40-244: ~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $ (lsb_release -cs) stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu noble stable'
Description:
Archive for codename: noble components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository...
Press [ENTER] to continue or Ctrl+C to cancel.
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Get:5 https://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:9 https://download.docker.com/linux/ubuntu/noble/stable amd64 Packages [15.3 kB]
Get:10 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
```

Run these commands to update the package list and install Docker:

sudo apt update

sudo apt install docker-ce docker-ce-cli containerd.io

```
ubuntu@ip-172-31-40-244: ~
nerd.service.
Setting up docker-compose-plugin (2.29.7-1~ubuntu.24.04~noble) ...
Setting up libltdl7:amd64 (2.4.7-7build1) ...
Setting up docker-ce-cli (5:27.3.1-1~ubuntu.24.04~noble) ...
Setting up libslirp0:amd64 (4.7.0-1ubuntu3) ...
Setting up pigz (2.8-1) ...
Setting up docker-ce-rootless-extras (5:27.3.1-1~ubuntu.24.04~noble) ...
Setting up slirp4netns (5.2.1-1build2) ...
Setting up docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.
No services need to be restarted.
No containers need to be restarted.
No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-40-244: ~
```

Next, run this command....

sudo mkdir -p /etc/docker

cat <<EOF | sudo tee /etc/docker/daemon.json

{

"exec-opts": ["native.cgroupdriver=systemd"]

}

EOF

```
sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
```

```
ubuntu@ip-172-31-40-244: $ sudo mkdir -p /etc/docker
ubuntu@ip-172-31-40-244: $ cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}
ubuntu@ip-172-31-40-244: $ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@ip-172-31-40-244: $ sudo systemctl daemon-reload
ubuntu@ip-172-31-40-244: $ sudo systemctl restart docker
```

Step 3: Kubernetes Installation

Run these commands to install kubernetes:

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

```
ubuntu@ip-172-31-40-244: $ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
ubuntu@ip-172-31-40-244: $ echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /" | Out-File -FilePath "C:\path\to\your\directory\kubernetes.list" -Encoding utf8
Out-File: command not found
ubuntu@ip-172-31-40-244: $ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
> https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/
ubuntu@ip-172-31-40-244: $ -
```

After this step, run the following commands,

```
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

Over here, while running sudo apt-get update if you encounter an error like this

```
ubuntu@ip-172-31-40-244:~$ sudo apt-get update
E: Malformed entry 1 in list file /etc/apt/sources.list.d/kubernetes.list (URI)
E: The list of sources could not be read.
```

Then run the following command and ensure that the file contains the following texts:

```
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
```

```
ubuntu@ip-172-31-40-244: ~
GNU nano 7.2
/etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb /
```

Be sure to save this file and then proceed with these commands

```
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

```
ubuntu@ip-172-31-40-244: ~
ubuntu@ip-172-31-40-244: $ sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/core:/stable:/v1.31/deb InRelease [1186 B]
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb Packages [4865 B]
Fetched 132 kB in 1s (199 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignored as the file has an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
ubuntu@ip-172-31-40-244: $ sudo apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 6 newly installed, 0 to remove and 142 not upgraded.
Need to get 87.4 MB of archives.
After this operation, 314 MB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 conntrack amd64 1:1.4.8-1ubuntu1 [37.9 kB]
```

```
Setting up conntrack (1:1.4.8-1ubuntu1) ...
Setting up kubelet (1.31.1-1.1) ...
Setting up cri-tools (1.31.1-1.1) ...
Setting up kubernetes-cni (1.5.1-1.1) ...
Setting up kubeadm (1.31.1-1.1) ...
Setting up kubelet (1.31.1-1.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-40-244: $ sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
ubuntu@ip-172-31-40-244: $
```

```
sudo systemctl enable --now kubelet
```

```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
```

```
ubuntu@ip-172-31-40-244:~$ sudo systemctl enable --now kubelet
ubuntu@ip-172-31-40-244: $ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
W0925 17:22:15.942659    6444 checks.go:1080] [preflight] WARNING: Couldn't create the interface used for talking to the container runtime: failed to create new CRI runtime service: validate service connection: validate CR
I vi runtime API for endpoint "unix:///var/run/containerd/containerd.sock": rpc error: code = Unimplemented desc =
c = unknown service runtime.v1.RuntimeService
      [WARNING FileExisting-socat]: socat not found in system path
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'
error execution phase preflight: [preflight] Some fatal errors occurred:
failed to create new CRI runtime service: validate service connection: validate CRI v1 runtime API for endpoint
"unix:///var/run/containerd/containerd.sock": rpc error: code = Unimplemented desc = unknown service runtime.v1.RuntimeService[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...
To see the stack trace of this error execute with --v=5 or higher
```

Since running that command ran us into an error, we are expected to install containerd first

Run this command: sudo apt-get install -y containerd

Then run,

```
sudo mkdir -p /etc/containerd
```

```
sudo containerd config default | sudo tee /etc/containerd/config.toml
```

```
ubuntu@ip-172-31-40-244: $ sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignored as the file has an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
ubuntu@ip-172-31-40-244: $ sudo apt-get install -y containerd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-ce-rootless-extras libltdl7 libslirp0 pigz slirp4netns
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  runc
The following packages will be REMOVED:
  containerd.io docker-ce
The following NEW packages will be installed:
  containerd runc
0 upgraded, 2 newly installed, 2 to remove and 142 not upgraded.
Need to get 47.2 MB of archives.
```

Post this step, run these commands one by one

```
sudo systemctl restart containerd
```

```
sudo systemctl enable containerd
```

```
sudo systemctl status containerd
```

```
ubuntu@ip-172-31-40-244:~$ sudo systemctl restart containerd
ubuntu@ip-172-31-40-244:~$ sudo systemctl enable containerd
ubuntu@ip-172-31-40-244:~$ sudo systemctl status containerd
● containerd.service - containerd container runtime
   Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
   Active: active (running) since Wed 2024-09-25 17:23:54 UTC; 17s ago
     Docs: https://containerd.io
     Main PID: 6555 (containerd)
        Tasks: 7
       Memory: 13.5M (peak: 14.1M)
         CPU: 121ms
        CGroup: /system.slice/containerd.service
                  └─6555 /usr/bin/containerd

Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231856399Z" level=info msg="Start"
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231887455Z" level=info msg=servin
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231896683Z" level=info msg="Start"
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231919602Z" level=info msg=servin
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231952908Z" level=info msg="Start"
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231972572Z" level=info msg="Start"
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231980403Z" level=info msg="Start"
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231986601Z" level=info msg="Start"
Sep 25 17:23:54 ip-172-31-40-244 systemd[1]: Started containerd.service - containerd container runtime.
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.234185443Z" level=info msg="conta
```

After this, run

```
sudo apt-get install -y socat
```

```
ubuntu@ip-172-31-40-244:~$ sudo apt-get install -y socat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-ce-rootless-extras libltdl7 libslirp0 pigz slirp4netns
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  socat
0 upgraded, 1 newly installed, 0 to remove and 142 not upgraded.
Need to get 374 kB of archives.
After this operation, 1649 kB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 socat amd64 1.8.0.0-4build3 [374 kB]
Fetched 374 kB in 0s (16.8 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68108 files and directories currently installed.)
Preparing to unpack .../socat_1.8.0.0-4build3_amd64.deb ...
Unpacking socat (1.8.0.0-4build3) ...
Setting up socat (1.8.0.0-4build3) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.
```

Step 4: Initialize kubecluster:

```
Sudo kubeadm init --pod-network-cidr=10.244.0.0/16
```

```
ubuntu@ip-172-31-40-244: ~
ubuntu@ip-172-31-40-244: $ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'
W0925 17:31:39.850175    7057 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the
container runtime is inconsistent with that used by kubeadm. It is recommended to use "registry.k8s.io/pause:3.
10" as the CRI sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [ip-172-31-40-244 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.40.244]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [ip-172-31-40-244 localhost] and IPs [172.31.40.244 12
7.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-40-244 localhost] and IPs [172.31.40.244 127.
0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
```

```
ubuntu@ip-172-31-40-244: ~
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate a
nd key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.40.244:6443 --token egcpta.ggcdfxh328wg5vm3 \
  --discovery-token-ca-cert-hash sha256:e392dc659374560eb409e01b69491d214ea39cb415ee934aaecefcff6a7f21d5
ubuntu@ip-172-31-40-244: $
```

Copy the mkdir and chown commands from the top and execute them.

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Add a common networking plugin called flannel as mentioned in the code.

```
kubectl apply -f
```

```
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
```

```
ubuntu@ip-172-31-40-244:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
ubuntu@ip-172-31-40-244:~$
```

Now that the cluster is up and running, we can deploy our nginx server on this Cluster. Apply this deployment file using this command to create a deployment

```
kubectl apply -f https://k8s.io/examples/application/deployment.yaml
```

```
ubuntu@ip-172-31-40-244:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-40-244:~$
```

kubectl get pods

```
ubuntu@ip-172-31-40-244:~$ kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-brgb7   0/1     Pending   0          48s
nginx-deployment-d556bf558-s5rtj   0/1     Pending   0          48s
ubuntu@ip-172-31-40-244:~$
```

```
POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath=".items[0].metadata.name")
```

```
kubectl port-forward $POD_NAME 8080:80
```

```
ubuntu@ip-172-31-40-244:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath=".items[0].metadata.name")
ubuntu@ip-172-31-40-244:~$ kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
ubuntu@ip-172-31-40-244:~$
```

To untaint all nodes, run the following command

```
kubectl taint nodes --all node-role.kubernetes.io/control-plane:NoSchedule-
```

Then run, kubectl get nodes

```
ubuntu@ip-172-31-40-244:~$ kubectl taint nodes --all node-role.kubernetes.io/control-plane:NoSchedule-
node/ip-172-31-40-244 untainted
ubuntu@ip-172-31-40-244:~$ kubectl get nodes
NAME      STATUS   ROLES      AGE   VERSION
ip-172-31-40-244   Ready   control-plane   16m   v1.31.1
ubuntu@ip-172-31-40-244:~$
```

Then run, kubectl get pods command again

```
ubuntu@ip-172-31-40-244:~$ kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-brgb7   1/1     Running   0          7m15s
nginx-deployment-d556bf558-s5rtj   1/1     Running   0          7m15s
ubuntu@ip-172-31-40-244:~$
```

Since, Jenkins server/service is already occupying port 8080, i tried starting the nginx service at 8082 port number. Following is the command that i executed was,
 kubectl port-forward \$POD_NAME 8082:80

```
ubuntu@ip-172-31-40-244:~$ kubectl port-forward svc/nginx 8082:80
Forwarding from 127.0.0.1:8082 -> 80
Forwarding from [::1]:8082 -> 80
Handling connection for 8082
```

The last step consisted of opening a new terminal and logging in using the SSH method that we used earlier and running the following command so as to finally confirm that we have successfully deployed our kubernetes application over the nginx server, using nginx service (optional)

curl --head http://127.0.0.1:8082

```
Last login: Wed Sep 25 19:13:52 2024 from 45.112.58.76
ubuntu@ip-172-31-40-244:~$ curl --head http://127.0.0.1:8082
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Wed, 25 Sep 2024 19:36:26 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 04 Dec 2018 14:44:49 GMT
Connection: keep-alive
ETag: "5c0692e1-264"
Accept-Ranges: bytes
```

Conclusion:

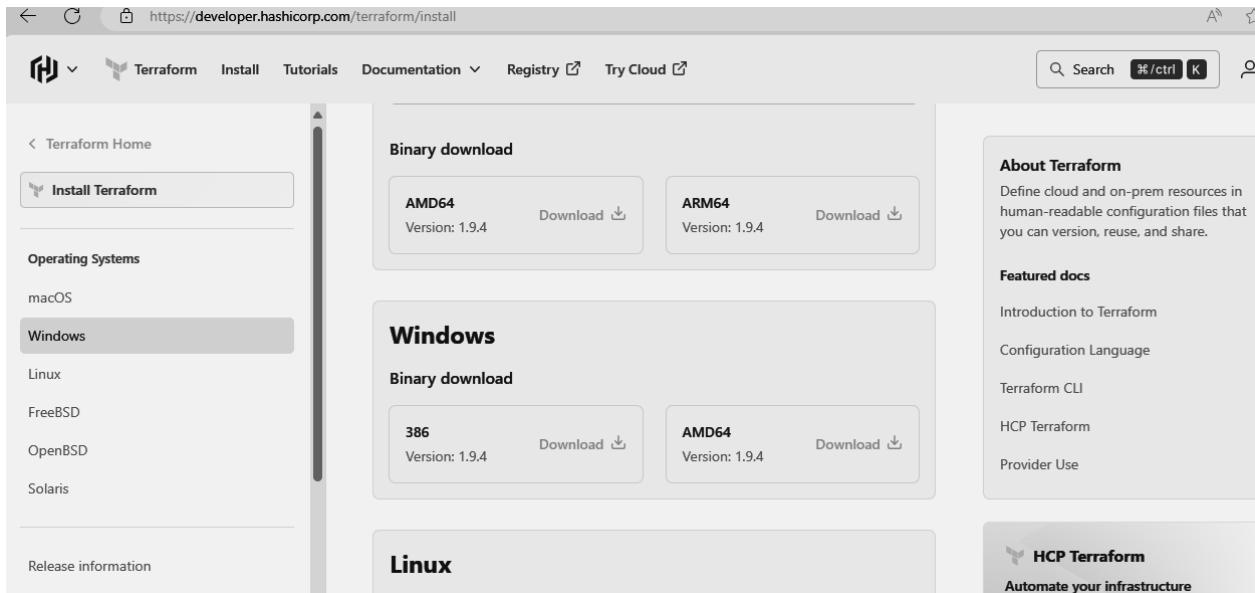
In this experiment, we successfully deployed an Nginx web server on a Kubernetes cluster and exposed it using NodePort services. By creating the deployment and configuring the service, we demonstrated Kubernetes' ability to manage containerized applications efficiently.

The use of 'kubectl' commands allowed us to check pod status and access the Nginx server locally via port forwarding on the EC2 instance. This experience provided valuable insights into service management and networking concepts within Kubernetes.

Overall, this experiment reinforced the fundamental skills needed for deploying cloud-native applications and laid the groundwork for exploring more advanced Kubernetes features in future projects.

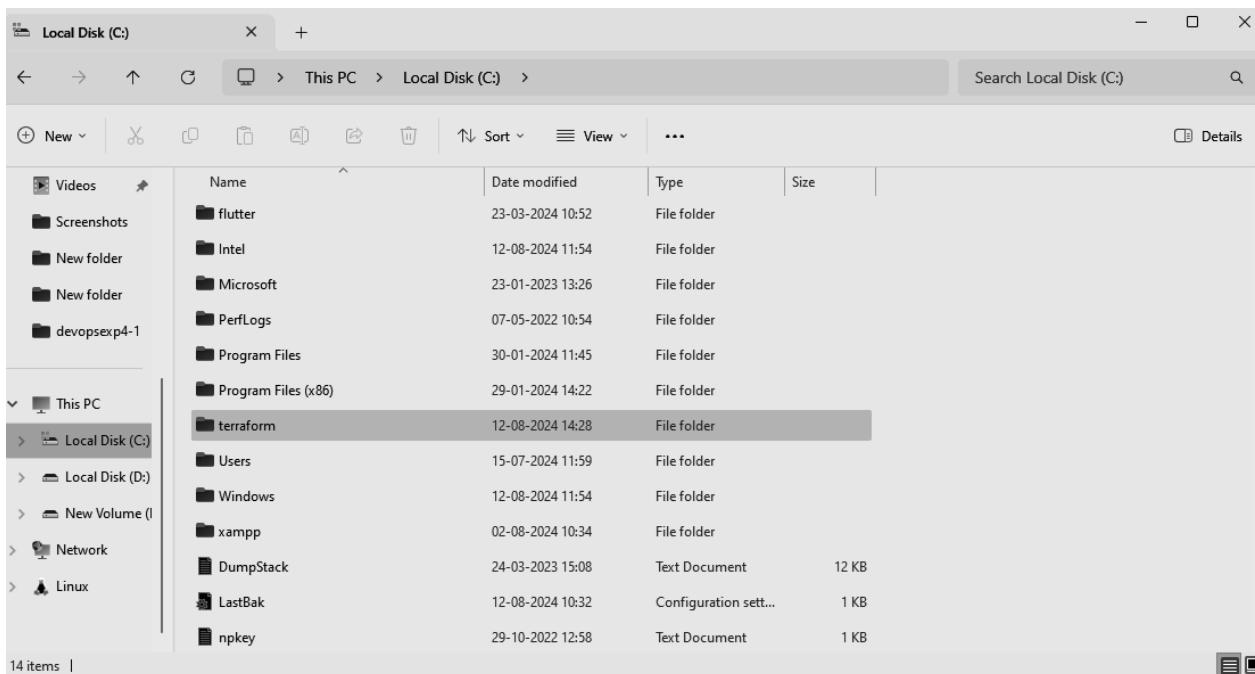
Exp 05: To understand terraform lifecycle, core concepts/terminologies and install it on a Linux Machine.

Step 1: Navigate to hashicorp.com page and download the AMD64 file for Terraform. Install it for Windows.



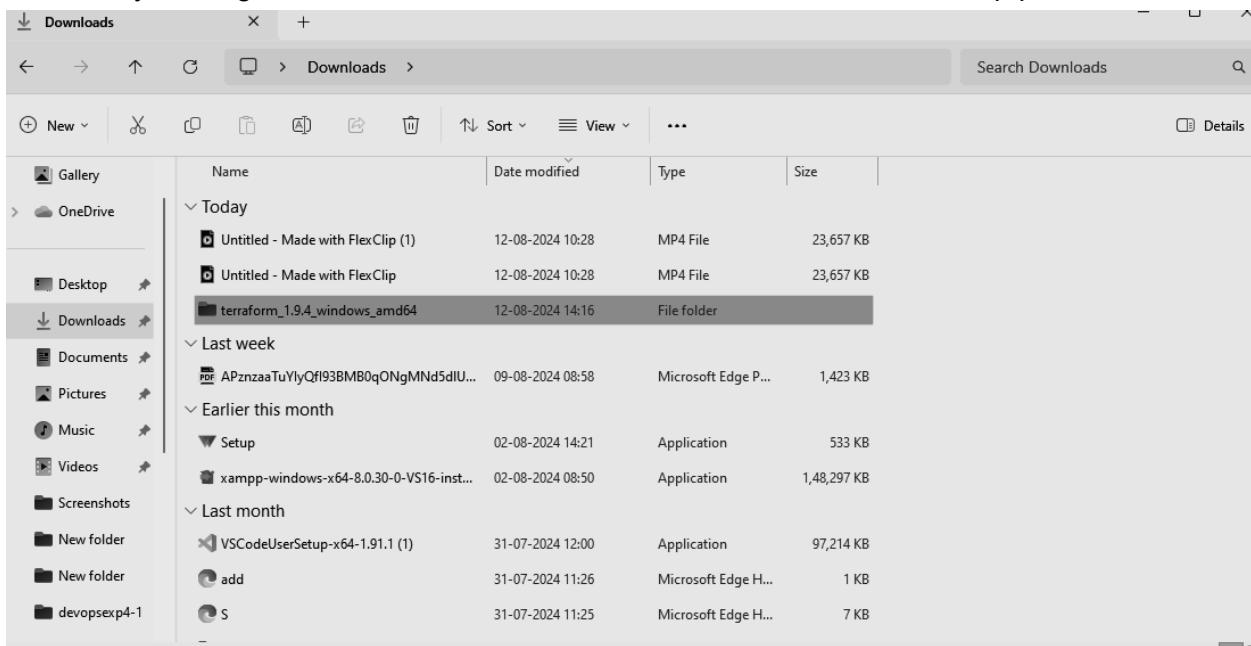
The screenshot shows the HashiCorp Terraform installation page. On the left, there's a sidebar for 'Operating Systems' with 'Windows' selected. The main area has sections for 'Binary download' under 'Windows' and 'Linux'. Under Windows, there are links for '386' (Version: 1.9.4) and 'AMD64' (Version: 1.9.4), each with a 'Download' button. To the right, there's a sidebar titled 'About Terraform' with links to various documentation and a 'HCP Terraform' section.

Step 2: Create a new folder inside your C drive named Terraform for your convenience. We would need it for setting up our PATH variable inside system variables.

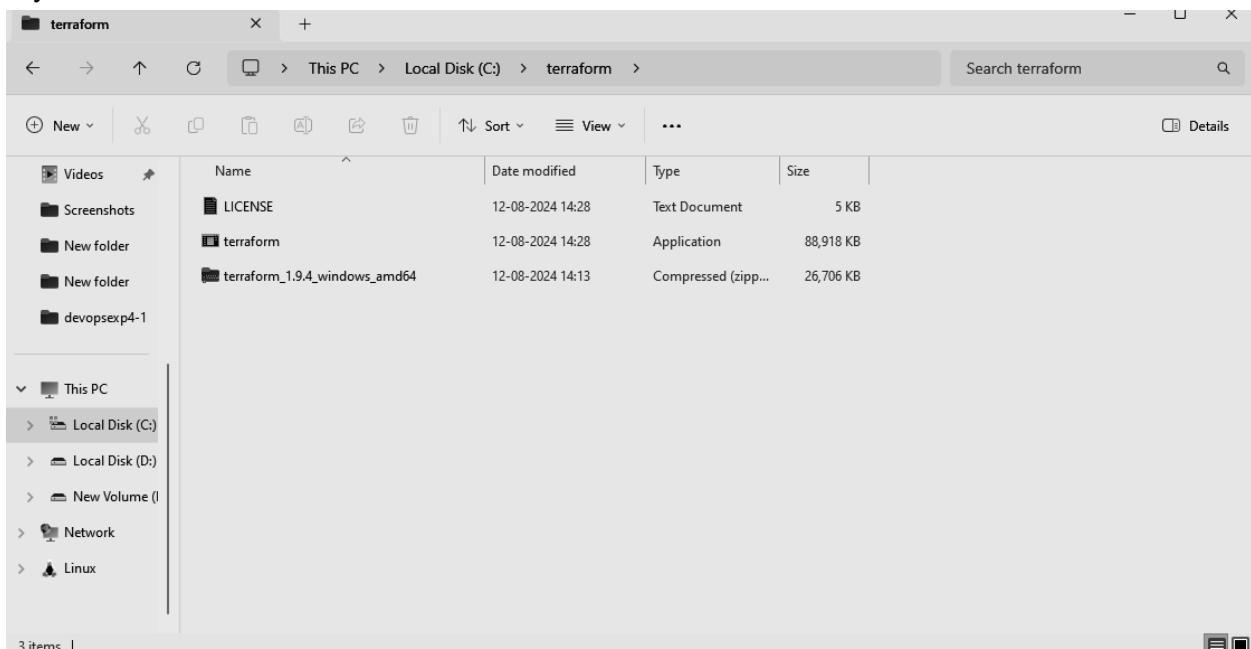


The screenshot shows a Windows File Explorer window. The left sidebar shows 'Local Disk (C:)'. The main area displays a list of files and folders. A new folder named 'terraform' is visible in the list, highlighted with a gray selection bar. The folder details are shown in the bottom right corner: Date modified: 12-08-2024 14:28, Type: File folder, Size: 12 KB.

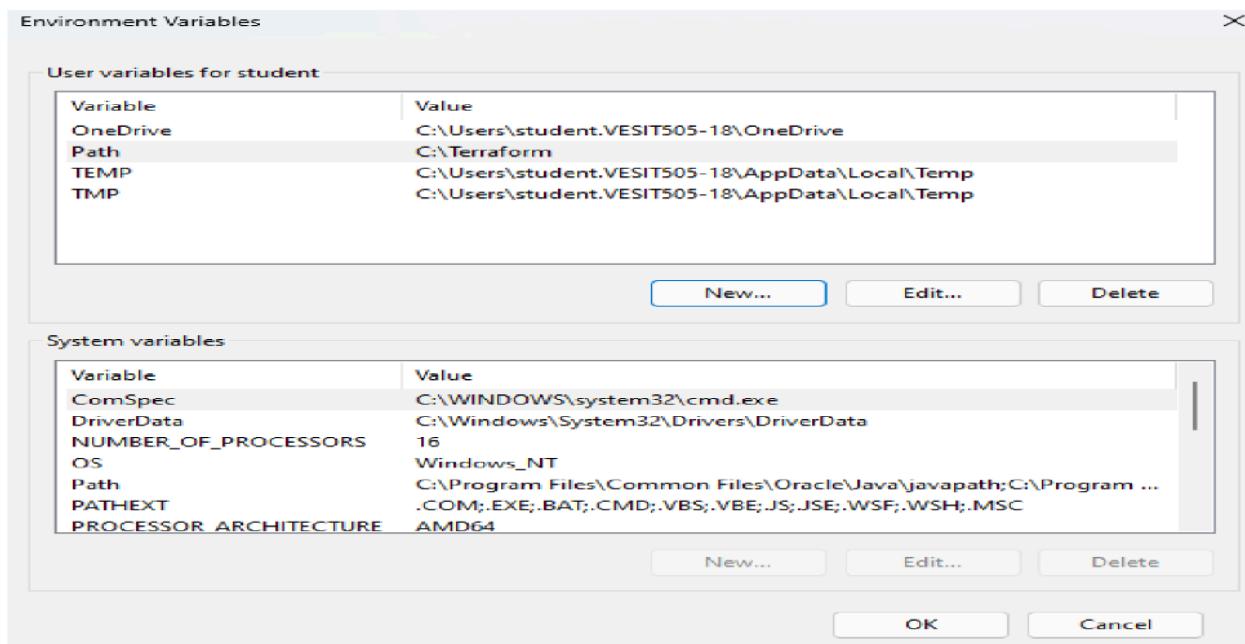
Step 3: Select the package that you have downloaded and extract the files within it. Set it in such a way that it gets extracted in the Terraform folder we created in the step prior to this one



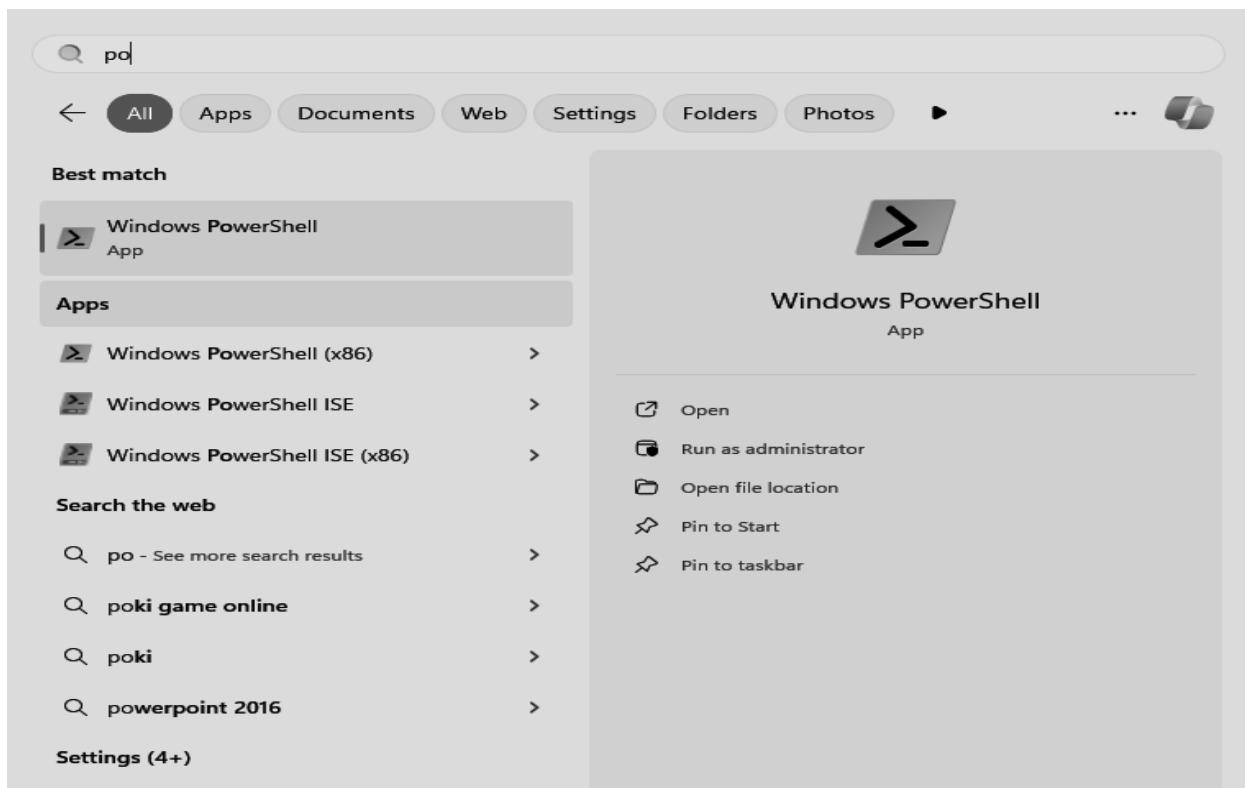
Step 4: Ensure that you have the following files appearing in your newly made Terraform folder in your local C drive



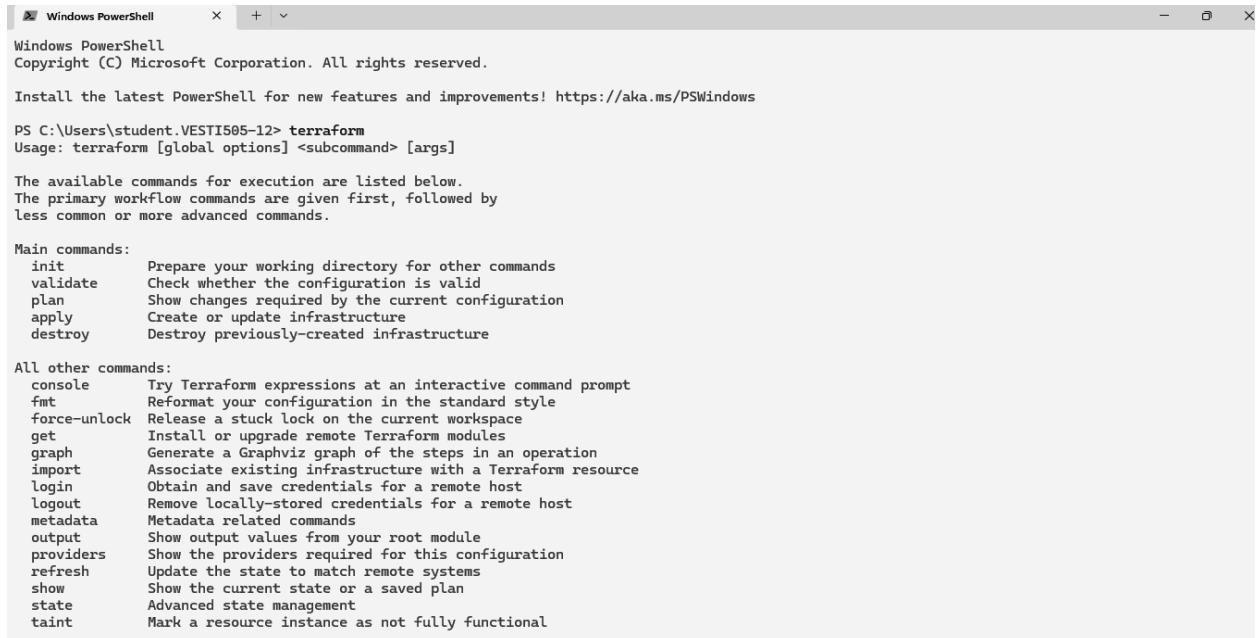
Step 5: Add the path of your Terraform folder in the PATH variable in the system environment variables inside settings.



Step 6: Open Windows PowerShell and select 'Run as Administrator' option for carrying out the next step



Step 7: After opening Windows PowerShell, type the command ‘terraform’. If it is properly extracted and path is added to system variables, then following is the result we would be able to see. It will show us numerous other commands and options that we can try out so as to work with Terraform



A screenshot of a Windows PowerShell window titled "Windows PowerShell". The window shows the output of the "terraform" command. It includes copyright information, a link to the latest PowerShell features, and usage instructions. It then lists available commands under two sections: "Main commands" and "All other commands", each with a brief description.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\student.VESTI505-12> terraform
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init      Prepare your working directory for other commands
  validate  Check whether the configuration is valid
  plan     Show changes required by the current configuration
  apply    Create or update infrastructure
  destroy   Destroy previously-created infrastructure

All other commands:
  console   Try Terraform expressions at an interactive command prompt
  fmt       Reformat your configuration in the standard style
  force-unlock Release a stuck lock on the current workspace
  get       Install or upgrade remote Terraform modules
  graph    Generate a Graphviz graph of the steps in an operation
  import   Associate existing infrastructure with a Terraform resource
  login    Obtain and save credentials for a remote host
  logout   Remove locally-stored credentials for a remote host
  metadata Metadata related commands
  output   Show output values from your root module
  providers Show the providers required for this configuration
  refresh  Update the state to match remote systems
  show     Show the current state or a saved plan
  state    Advanced state management
  taint    Mark a resource instance as not fully functional
```

Conclusion: In conclusion, this experiment walks through the process of installing and setting up Terraform on a Windows machine. By downloading the appropriate package from HashiCorp, extracting it into a designated folder, and adding the folder to the system's PATH variable, we can successfully configure Terraform for use. After completing the installation, running the 'terraform' command in PowerShell confirms that the setup is successful, providing access to various Terraform commands. This sets the foundation for managing infrastructure as code using Terraform on a Windows system.

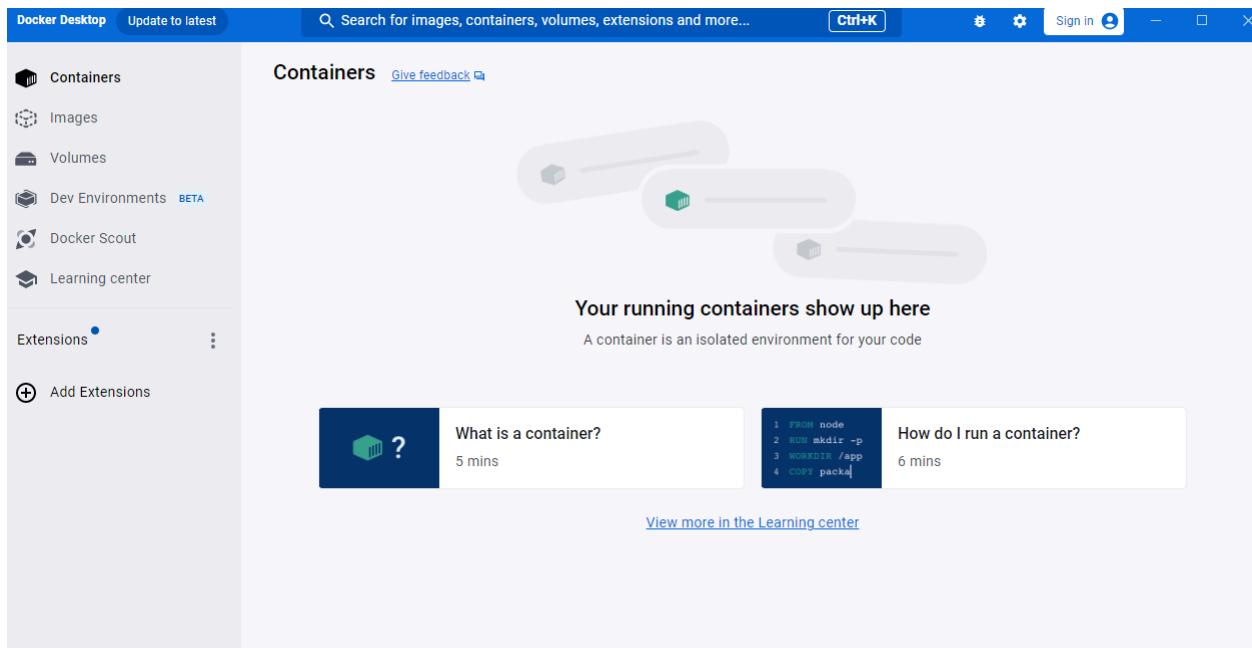
Adv DevOps Exp 06

Aim: To create docker image using terraform

Step 1: Check the functionality of docker desktop i.e Check if Docker is installed properly on your system and that Docker Desktop is opening properly (virtualization settings are needed to be enabled in our system bios)

```
C:\Users\INFT505-16>docker -v
Docker version 24.0.6, build ed223bc

C:\Users\INFT505-16>
```



Step 2: Now, create a folder named ‘Terraform Scripts’ in which we save our different types of scripts which will be further used in this experiment.

Step 2: Firstly create a new folder named ‘Docker’ in the ‘TerraformScripts’ folder. Then create a new docker.tf file using Atom editor and write the following contents into it to create a Ubuntu Linux container.

Script:

terraform

```
{ required_providers
{docker = {
source = "kreuzwerker/docker"
version = "2.21.0"
```

```
}
```

```
}
```

```
}
```

```
provider "docker" {
```

```
host = "npipe://./pipe/docker_engine"
```

```
}
```

```
# Pulls the image
```

```
resource "docker_image" "ubuntu"
```

```
{name = "ubuntu:latest"
```

```
}
```

```
# Create a container
```

```
resource "docker_container" "foo"
```

```
{ image =
```

```
docker_image.ubuntu.image_idname =
```

```
"foo"
```

```
}
```

```
docker.tf  X
```

```
Docker > docker.tf
```

```
1  terraform {
```

```
2    required_providers {
```

```
3      docker = {
```

```
4        source  = "kreuzwerker/docker"
```

```
5        version = "2.21.0"
```

```
6      }
```

```
7    }
```

```
8  }
```

```
9
```

```
10 provider "docker" {
```

```
11   host = "npipe://./pipe/docker_engine"
```

```
12 }
```

```
13
```

```
14 # Pull the image
```

```
15 resource "docker_image" "ubuntu" {
```

```
16   name = "ubuntu:latest"
```

```
17 }
```

```
18
```

```
19 # Create a container
```

```
20 resource "docker_container" "foo" {
```

```
21   image = docker_image.ubuntu.image_id
```

```
22   name  = "foo"
```

```
23   command = ["sleep", "3600"]
```

```
24 }
```

```
25
```

Step 3: Run terraform init command

Run the command **terraform init** in your terminal. This command initializes the Terraform configuration, downloads the necessary provider plugins specified in the `docker.tf` file, and prepares the working directory for further commands. This step is crucial to ensure that Terraform can interact with Docker.

```
C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of kreuzwerker/docker from the dependency lock file
- Using previously-installed kreuzwerker/docker v2.21.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>
```

Step 4: Run terraform plan command:

Next, execute the command **terraform plan** to create an execution plan. This command will show what actions Terraform will take based on your configuration. It ensures that the planned changes match your expectations before actual implementation.

```
C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
+ create

Terraform will perform the following actions:

# docker_container.foo will be created
+ resource "docker_container" "foo" {
    + attach          = false
    + bridge          = (known after apply)
    + command         = [
        + "sleep",
        + "3600",
    ]
    + container_logs = (known after apply)
    + entrypoint      = (known after apply)
    + env             = (known after apply)
    + exit_code       = (known after apply)
    + gateway         = (known after apply)
    + hostname        = (known after apply)
    + id              = (known after apply)
    + image           = (known after apply)
    + init            = (known after apply)
    + ip_address      = (known after apply)
    + ip_prefix_length = (known after apply)
    + ipc_mode        = (known after apply)
    + log_driver      = (known after apply)
    + logs            = false
    + must_run        = true
    + name            = "foo"
    + network_data    = (known after apply)
    + read_only       = false
    + remove_volumes = true
}
```

```

+ rm          = false
+ runtime     = (known after apply)
+ security_opts = (known after apply)
+ shm_size    = (known after apply)
+ start       = true
+ stdin_open   = false
+ stop_signal  = (known after apply)
+ stop_timeout = (known after apply)
+ tty         = false

+ healthcheck (known after apply)

+ labels (known after apply)
}

# docker_image.ubuntu will be created
+ resource "docker_image" "ubuntu" {
  + id      = (known after apply)
  + image_id = (known after apply)
  + latest   = (known after apply)
  + name     = "ubuntu:latest"
  + output   = (known after apply)
  + repo_digest = (known after apply)
}

Plan: 2 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>

```

Step 5: Execute terraform apply command:

Run **terraform apply** to apply the configuration and create the specified resources. During this step, Terraform will prompt you to confirm the action by typing yes. Upon confirmation, Terraform will pull the Ubuntu image and create the container, reflecting successful deployment.

```

C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with:
+ create

Terraform will perform the following actions:

# docker_container.foo will be created
+ resource "docker_container" "foo" {
  + attach          = false
  + bridge          = (known after apply)
  + command         = [
      + "sleep",
      + "3600",
    ]
  + container_logs  = (known after apply)
  + entrypoint      = (known after apply)
  + env              = (known after apply)
  + exit_code        = (known after apply)
  + gateway          = (known after apply)
  + hostname         = (known after apply)
  + id               = (known after apply)
  + image             = (known after apply)
  + init              = (known after apply)
  + ip_address        = (known after apply)
  + ip_prefix_length = (known after apply)
  + ipc_mode          = (known after apply)
  + log_driver        = (known after apply)
  + logs              = false
  + must_run          = true
  + name              = "foo"
  + network_data      = (known after apply)
  + read_only          = false
  + remove_volumes   = true
}

```

We are supposed to put yes to proceed ahead

```

+ rm          = false
+ runtime     = (known after apply)
+ security_opts = (known after apply)
+ shm_size    = (known after apply)
+ start       = true
+ stdin_open   = false
+ stop_signal  = (known after apply)
+ stop_timeout = (known after apply)
+ tty          = false

+ healthcheck (known after apply)

+ labels (known after apply)
}

# docker_image.ubuntu will be created
+ resource "docker_image" "ubuntu" {
  + id      = (known after apply)
  + image_id = (known after apply)
  + latest   = (known after apply)
  + name     = "ubuntu:latest"
  + output   = (known after apply)
  + repo_digest = (known after apply)
}

```

Plan: 2 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

After executing terraform apply:

```

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

docker_image.ubuntu: Creating...
docker_image.ubuntu: Still creating... [10s elapsed]
docker_image.ubuntu: Creation complete after 12s [id=sha256:b1e9cef3f2977f8bdd19eb9ae04f83b315f80fe4f5c5651fedf41482c12432f7ubuntu:latest]
docker_container.foo: Creating...
docker_container.foo: Creation complete after 1s [id=684alde22619abba495d0c2886fc9d8f681302574f3e1ca16c5297470b4e88b4]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>

```

Terraform images after apply step:

```

C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
ubuntu          latest   b1e9cef3f297    3 weeks ago   78.1MB
sonarqube       latest   3183d6818c6e  11 months ago  716MB

C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>

```

Step 6: Run terraform destroy command

To remove the resources created during the experiment, run the command **terraform destroy**.

This command will prompt for confirmation before proceeding to delete the container and any other associated resources. It's essential for maintaining a clean working environment and avoiding unnecessary resource consumption.

```
C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:b1e9cef3f2977f8bdd19eb9ae04f83b315f80fe4f5c5651fedf41482c12432f7ubuntu:latest]
docker_container.foo: Refreshing state... [id=684a1de22619abba495d0c2886fc9d8f681302574f3e1ca16c5297470b4e88b4]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
- destroy

Terraform will perform the following actions:

# docker_container.foo will be destroyed
resource "docker_container" "foo" {
  - attach           = false -> null
  - command          = [
    - "sleep",
    - "3600"
  ] -> null
  - cpu_shares       = 0 -> null
  - dns              = [] -> null
  - dns_opts         = [] -> null
  - dns_search        = [] -> null
  - entrypoint        = [] -> null
  - env              = [] -> null
  - gateway          = "172.17.0.1" -> null
  - group_add        = [] -> null
  - hostname          = "684a1de22619" -> null
  - id               = "684a1de22619abba495d0c2886fc9d8f681302574f3e1ca16c5297470b4e88b4" -> null
  - image             = "sha256:b1e9cef3f2977f8bdd19eb9ae04f83b315f80fe4f5c5651fedf41482c12432f7" -> null
  - init              = false -> null
  - ip_address        = "172.17.0.2" -> null
  - ip_prefix_length  = 16 -> null
  - ipc_mode          = "private" -> null
  - links             = [] -> null
  - log_driver         = "json-file" -> null
  - log_opts           = {} -> null
}
```

After executing terraform destroy command:

```
C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
sonarqube      latest   3183d6818c6e   11 months ago   716MB
```

```
C:\Users\INFT505-16\Desktop\TerraformScripts\Docker>|
```

Conclusion: In this experiment, we effectively utilized Terraform to manage Docker resources by creating an organized folder structure and executing a series of commands to deploy an Ubuntu container. By following the steps from initialization to resource destruction, we gained hands-on experience in infrastructure as code (IaC) principles, enhancing our understanding of managing containers with Terraform and Docker. This practical approach not only solidified our knowledge but also emphasized the importance of automated resource management in modern DevOps practices.

Adv DevOps Exp 07

Aim: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

Step 1:

Ensure Installations and path settings of the following softwares/services:

1. Jenkins
2. Docker Desktop Engine

Both these services help us create a sonarqube image and build a project within sonarqube using the pipeline architecture in jenkins.

Run this command in powershell administrator mode: docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest

This kickstarts the Sonarqube service/server at port **9000**. It also creates a Docker image within our Docker Desktop account. Make sure to keep your Docker Engine up and running

Powershell output:

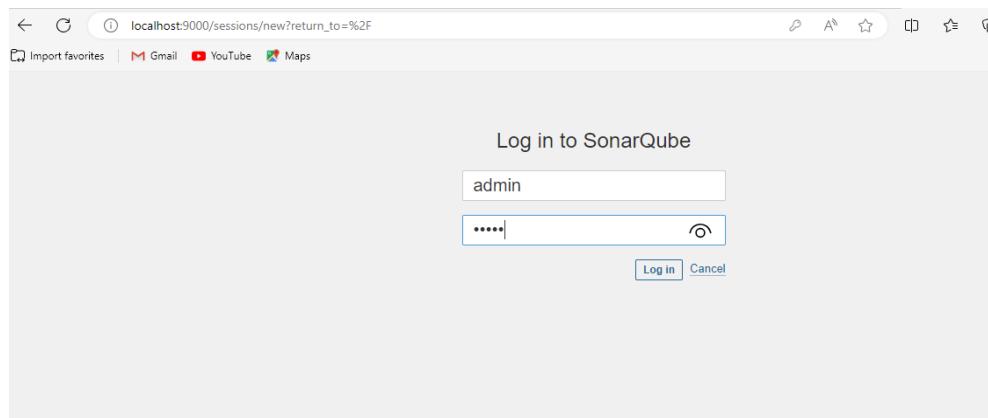
```
PS C:\Users\INFT505-16> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
2d9047cbfbab55bd03ed72f7349feebf5d79434b23ff7eda329694bd24115349
PS C:\Users\INFT505-16> |
```

Sonarqube started on Docker:

	Name	Image	Status	CPU (%)	Port(s)	Last started
	 sonarqube 2d9047cbfbab55bd03ed72f7349feebf5d79434b23ff7eda329694bd24115349	sonarqube:latest	Running	282.07%	9000:9000	12 seconds ago

Go to localhost:9000. On fulfilling the desired action, we arrive on a page that asks us for our username and password.

The default credentials for Sonarqube server are (username,password) == (**admin,admin**)



localhost:9000/sessions/new?return_to=%2F

Import favorites | Gmail | YouTube | Maps

Log in to SonarQube

admin	
.....	⊕
<input type="button" value="Log in"/>	<input type="button" value="Cancel"/>

Sonarqube asks us to update the default password and make a new password for this Sonarqube account. So i entered a new suitable password (keep it easy enough for one to remember)

The screenshot shows a web browser window with the URL `localhost:9000/account/reset_password`. The page title is "Update your password". A message at the top states: "This account should not use the default password." Below this, there is a form with three fields: "Old Password *", "New Password *", and "Confirm Password *". Each field has a placeholder with four dots. A blue "Update" button is located at the bottom right of the form area.

Step 2:

After logging in,

This is the page that we land on... This tells us that with the help of docker we have successfully created a Sonarqube image and that we have assigned a port number 9000 to it so as to use its services and build a Sonarqube project.

From here on, we are supposed to make a new project manually by selecting the 'Create project manually' option

The screenshot shows the SonarQube dashboard. At the top, there is a notification bar: "There's a new version of SonarQube available. Update to enjoy the latest updates and features." with a "Learn More" link. Below the header, there are navigation links: Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, More, and a search icon. A sidebar on the left says "Create your project from your favorite DevOps platform." It lists several import options: "Import from Azure DevOps" (Setup), "Import from Bitbucket Cloud" (Setup), "Import from Bitbucket Server" (Setup), "Import from GitHub" (Setup), and "Import from GitLab" (Setup). At the bottom of the sidebar, there is a link "Create project manually". A yellow warning box at the bottom states: "Embedded database should be used for evaluation purposes only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine." The footer contains the text: "SonarQube™ technology is powered by SonarSource SA. Community Edition - Version 10.2.1 (build 78527) - LGPLv3 - Community - Documentation - Plugins - Web API".

We are now supposed to enter the name for our Sonarqube
Let's keep it as 'sonarqube'

Create a project

Project display name *

Up to 255 characters. Some scanners might override the value you provide.

Project key *

The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

Main branch name *

The name of your project's default branch [Learn More](#)

Next

Next, we are asked for setting up our sonarqube project
We will select global settings for this project and finally create the project named 'sonarqube'

Set up project for Clean as You Code

The new code definition sets which part of your code will be considered new code. This helps you focus attention on the most recent changes in your project, enabling you to follow the Clean as You Code methodology. Learn more: [Defining New Code](#)

Choose the baseline for new code for this project

Use the global setting

Previous version

Any code that has changed since the previous version is considered new code.
Recommended for projects following regular versions or releases.

Define a specific setting for this project

Previous version

Any code that has changed since the previous version is considered new code.
Recommended for projects following regular versions or releases

Step 3: Go to Jenkins Dashboard and download the plugin named '**Sonarqube Scanner for Jenkins**'. This plugin will help us to build our Sonarqube project through a project created within Jenkins. Make sure to restart Jenkins once installation is complete and Enable this plugin to use its services

Plugins

Updates: 57

Available plugins

Installed plugins: SonarQube Scanner for Jenkins 2.17.2 (Enabled)

Advanced settings

Search bar: sonar

Description: This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.

Enable checkbox: checked

Go to jenkins system configurations, and navigate to Sonarqube installation
 Enter the Sonarqube's project name, the URL and server authentication token if needed

Dashboard > Manage Jenkins > System >

SonarQube installations
List of SonarQube installations

Name
sonarqube

Server URL
Default is http://localhost:9000
http://localhost:9000/tutorials?id=sonarqube

Server authentication token
SonarQube authentication token. Mandatory when anonymous access is disabled.
- none -
+ Add ▾

Save Apply

Select 'install automatically' option for installation of the specified Sonarqube Scanner version

Install automatically ?

Install from Maven Central

Version
SonarQube Scanner 6.1.0.4477

Add Installer ▾

Add SonarQube Scanner

Save Apply

Step 4: Make a freestyle project in jenkins and give it a suitable name. This project is being made by us, so as to carry out our build of our Sonarqube project and perform static analysis of it based on whether the build is successful or not.

Jenkins

Search (CTRL+K) ? ⚙

Dashboard > All > New Item

New Item

Enter an item name
project1

Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

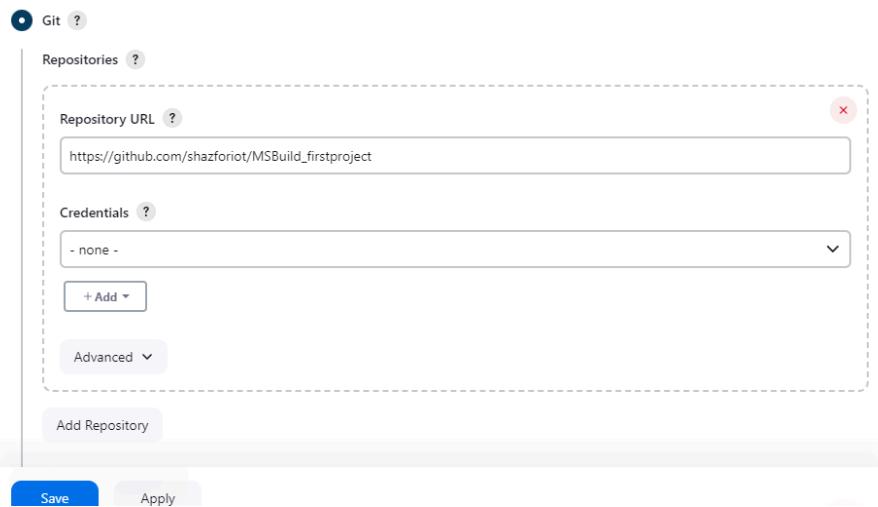
Multi-configuration project
Creates parallel builds for variants that need a large number of different configurations, such as testing on multiple environments.

OK

Choose this GitHub repository in Source Code Management.

https://github.com/shazforiot/MSBuild_firstproject.git

It is a sample hello-world project with no vulnerabilities and issues, just to test



Next, under build steps, select 'Execute Sonarqube Scanner' option

The screenshot shows the Jenkins 'Configuration' screen for a project named 'project1'. The left sidebar lists 'General', 'Source Code Management', 'Build Triggers', 'Build Environment' (which is selected), 'Build Steps', and 'Post-build Actions'. In the main area, under 'Build Steps', a dropdown menu titled 'Add build step' is open, showing options like 'Execute SonarQube Scanner', 'Execute Windows batch command', 'Execute shell', etc. The 'Execute SonarQube Scanner' option is highlighted.

There is an option for entering 'Analysis properties' in which i entered the following credentials/parameters related to my Sonarqube account or the project that i have made:

```
sonar.projectKey=com.example.myproject
sonar.projectName=My Sample Project
sonar.projectVersion=1.0
sonar.sources=.
sonar.host.url=http://localhost:9000
sonar.login=sqa_180e0f1309adfbba226d862355a9feb1066eddf
sonar.projectBaseDir=C:/ProgramData/Jenkins/.jenkins/workspace/project1
```

The screenshot shows the Jenkins 'Configuration' page for a project named 'project1'. On the left, there's a sidebar with options: General, Source Code Management, Build Triggers, Build Environment, Build Steps (which is selected), and Post-build Actions. In the main area, under 'Analysis properties', there is a text input field containing the analysis properties listed above. Below it, there are sections for 'Additional arguments' and 'JVM Options', each with their own text input fields.

Step 5: In the analysis properties, in order to set the **sonar.login** parameter, we are supposed to navigate to our account details and choose the option of generating a global token. Create a new token...

Now, this assists us to identify our project uniquely without having to provide our credentials or in case our credentials fail to be identified

This token creation step is optional and is only to be used when our analysis properties consisting of our normal credentials fail

The screenshot shows the 'Tokens' management interface in SonarQube. At the top, there's a note about using tokens for security instead of user logins. Below it, there's a 'Generate Tokens' form with fields for 'Name' (Enter Token Name), 'Type' (Select Token Type dropdown with '30 days'), and 'Expires in' (30 days). A 'Generate' button is next to the expiration field. Below this, a table lists existing tokens: 'token1' (Global type, last used < 1 hour ago, created September 23, 2024, expired October 23, 2024, with a 'Revoke' button). At the bottom, there's a placeholder 'Enter a new password'.

Now, navigate to security within administration and provide administrator system and execute analysis rights to the entity named admin.

All	Users	Groups	Search for users or groups...	Administer System	Administrator	Execute Analysis	Create
				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Quality Gates <input checked="" type="checkbox"/> Quality Profiles	<input type="checkbox"/>	<input checked="" type="checkbox"/> Projects
				<input type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Projects
				<input type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input type="checkbox"/>	<input type="checkbox"/> Projects
				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Quality Gates <input checked="" type="checkbox"/> Quality Profiles	<input checked="" type="checkbox"/>	<input type="checkbox"/> Projects

4 of 4 shown

After Building...the console output should indicate success

This indicates that the image that we created in docker successfully led us to creating a project within sonarqube, which in turn led us to building it over jenkins by uniquely identifying our sonarqube project. Build stage of our sonarqube project in jenkins involves static analysis for which we passed in the appropriate properties

Status: #6 (Sep 23, 2024, 2:24:02 PM)

Started by anonymous user

Revision: f2bc042c04c6e72427c380bcae6d6fee7b49adf

Repository: https://github.com/shazforiot/MSBuild_firstproject

No changes.

Conclusion: The integration of Jenkins with SonarQube for static analysis using Docker provides a streamlined and automated approach to ensuring code quality. By successfully configuring the necessary tools—Docker for hosting SonarQube and Jenkins for building and analyzing the project—you can perform seamless static analysis using SonarQube's powerful scanning capabilities. This setup automated vulnerability detection, ensuring that potential issues are caught early. The use of GitHub as a repository for code allows for real-time scanning and continuous integration, ensuring that projects meet quality standards before moving forward in the development lifecycle.

Adv DevOps Exp 08

Aim: Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.

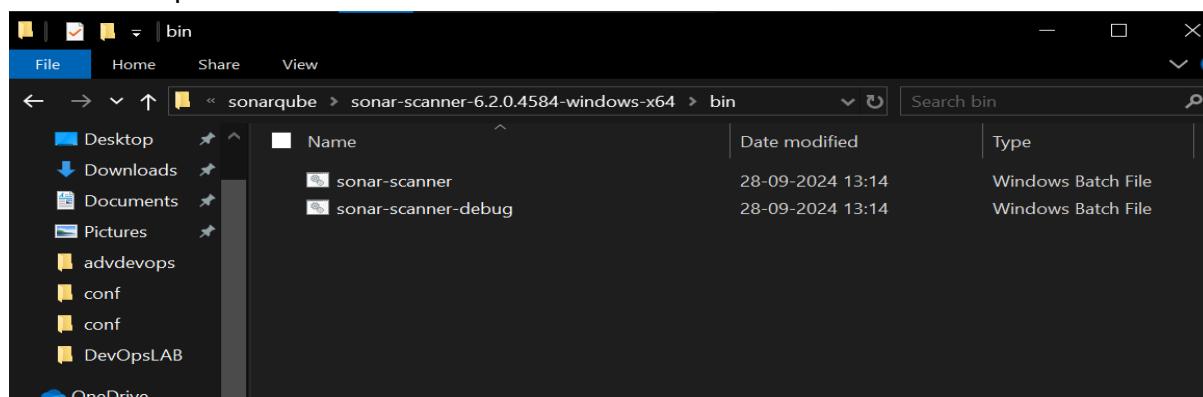
(I have performed this experiment entirely on my laptop)

For the Docker installation problems that I was facing during experiments 6 and 7, I resorted to the option of Resetting Windows which helped me clear the corrupted files which were causing my Laptop's Restart issues

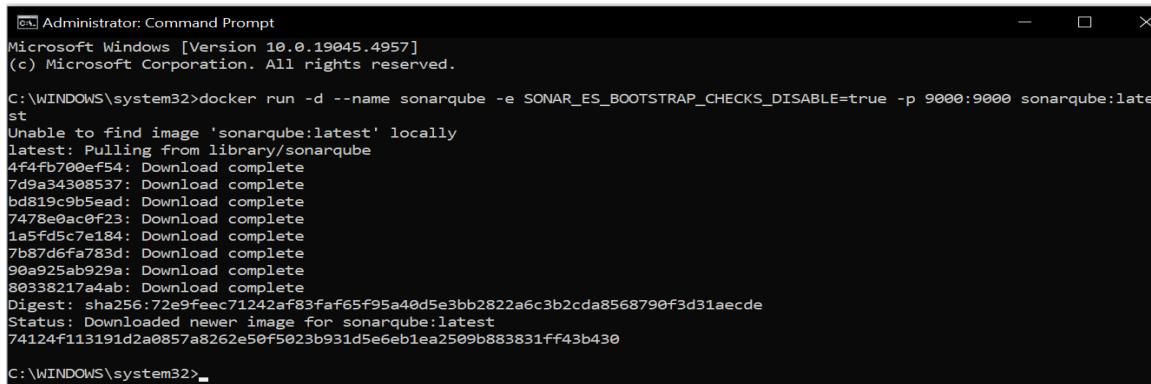
Step 1: Download sonar scanner. We will require it in the further steps while building our pipeline [SonarScanner CLI \(sonarsource.com\)](https://www.sonarsource.com/products/sonar-scanner/)

Visit this link and download the sonarqube scanner CLI.

Extract the zip files in a folder



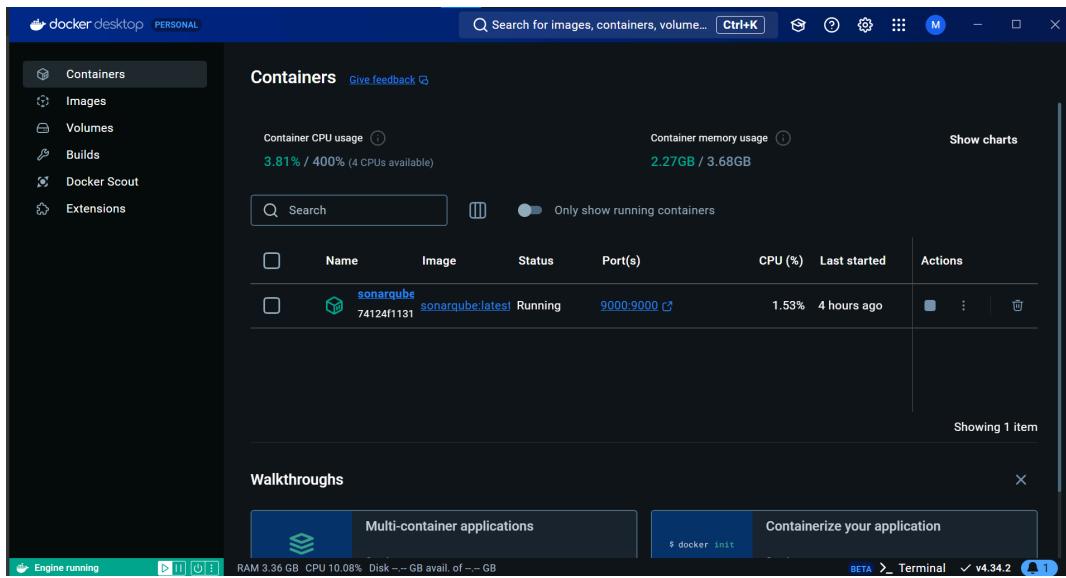
Run this command in an administrative command prompt so as to create a sonarqube image on localhost:9000



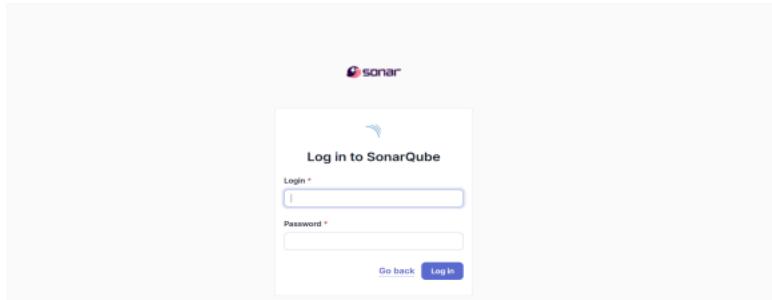
```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19045.4957]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
st
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
4f4fb700ef54: Download complete
7d9a34308537: Download complete
bd8b19c9b5ead: Download complete
7478e0ac0f23: Download complete
1a5fd5c7e184: Download complete
7b87d6fa783d: Download complete
90a925ab929a: Download complete
80338217a4ab: Download complete
Digest: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cd8568790f3d31aecde
Status: Downloaded newer image for sonarqube:latest
74124f113191d2a0857a8262e5023b931d5e6eb1ea2509b883831ff43b430

C:\WINDOWS\system32>
```



Visit localhost:9000 on your browser and login to your account(the default credentials are admin and admin for username and password. It allows to change it and set our own password for our account)



Next, we are supposed to Create a project. Give some suitable name to your project. A project will be generated for it automatically. We can change it or keep it the same

sonarqube

Projects Issues Rules Quality Profiles Quality Gates

Create a project

Project display name *

Up to 255 characters. Some scanners might override the value you provide.

Project key *

The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

Main branch name *

The name of your project's default branch [Learn More](#)

Next

Choose global settings option for setting up your project

sonarqube

Projects Issues Rules Quality Profiles Quality Gates Administration More Q

Set up project for Clean as You Code

The new code definition sets which part of your code will be considered new code. This helps you focus attention on the most recent changes in your project, enabling you to follow the Clean as You Code methodology. Learn more: [Defining New Code](#)

Choose the baseline for new code for this project

Use the global setting

Previous version

Any code that has changed since the previous version is considered new code.
Recommended for projects following regular versions or releases.

Define a specific setting for this project

Previous version

Any code that has changed since the previous version is considered new code.
Recommended for projects following regular versions or releases

Now, on jenkins ..

For a few contents inside the script to be built using our pipeline, we need to configure those changes inside the global configuration settings inside Manage Jenkins.

First, Go to plugins inside Manage Jenkins and install the Sonarqube Scanner plugin which is necessary for the build to be executed.

The screenshot shows the Jenkins Plugins page. A search bar at the top contains the text "sonar". Below it, a sidebar on the left lists "Updates", "Available plugins", "Installed plugins" (which is selected and highlighted in grey), and "Advanced settings". The main content area displays a table with one row for the "SonarQube Scanner for Jenkins" plugin, version 2.17.2. The table columns are "Name" and "Enabled". The plugin is listed as "Enabled" with a green checkmark icon and a red "x" icon for unenabling.

Next, we will have to add our Sonarqube project specific credentials in the Credentials section of manage jenkins

The screenshot shows the Jenkins Credentials configuration page. The URL is "Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) > admin/***** (made for sonarqube)". The page has fields for "Scope" (set to "Global (Jenkins, nodes, items, all child items, etc)"), "Username" (set to "admin"), and "Password" (set to "Concealed"). There is also a "Treat username as secret" checkbox. Other fields include "ID" (set to "668f12f9-47dd-46ee-85ba-232bfc1e3865") and "Description" (set to "made for sonarqube"). A blue "Save" button is at the bottom.

Here, i added the username i.e admin and my password ... for which jenkins generated a automated ID for the credential block.

Now, our pipeline script will contain a line where we are supposed to provide the build with our source directory of the sonar-scanner batch file which we installed in the initial steps of our experiment

C:\sonarqube\sonar-scanner-6.2.0.4584-windows-x64\bin\sonar-scanner.bat

We will add this path inside the script to be built

Now, come to the dashboard

Create new item and select pipeline project. Give it a suitable name

The screenshot shows the Jenkins 'New Item' creation interface. At the top, there's a search bar with the placeholder 'Search (CTRL+K)' and a help icon. Below the header, the breadcrumb navigation shows 'Dashboard > All > New Item'. The main section is titled 'New Item' and contains a text input field labeled 'Enter an item name' with the value 'exp-08'. Below this, there's a section titled 'Select an item type' with three options: 'Freestyle project', 'Pipeline', and 'Multi-configuration project'. The 'Pipeline' option is highlighted with a blue background and a description: 'Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.' A blue 'OK' button is located at the bottom of this section.

Put this script in the Pipeline Definition section and select the mode as Pipeline script

```
node {
    stage('Cloning the GitHub Repo') {
        git 'https://github.com/shazforiot/GOL.git'
    }

    stage('SonarQube analysis') {
        withSonarQubeEnv('sonarqube') {
            withCredentials([usernamePassword(credentialsId:
                '668f12f9-47dd-46ee-85ba-232bfc1e3865', usernameVariable: 'SONAR_USER',
                passwordVariable: 'SONAR_PASSWORD')]) {
                bat """
                    C:/sonarqube/sonar-scanner-6.2.0.4584-windows-x64/bin/sonar-scanner.bat ^
                    -D sonar.login=%SONAR_USER% ^
                    -D sonar.password=%SONAR_PASSWORD% ^
                    -D sonar.projectKey=exp-08 ^
                    -D sonar.exclusions=vendor/**,resources/**,*/*.java ^
                    -D sonar.host.url=http://127.0.0.1:9000
                """
            }
        }
    }
}
```

```

1 node {
2   stage('Cloning the GitHub Repo') {
3     git 'https://github.com/shazforiot/GOL.git'
4   }
5
6   stage('SonarQube analysis') {
7     withSonarQubeEnv('sonarqube') {
8       withCredentials([usernamePassword(credentialsId: '668f12f9-47dd-46ee-85ba-232bf1e3865', usernameVariable: 'SONAR_USER', passwordVariable: 'SONAR_PASSWORD')])
9       bat """
10         C:/sonarqube/sonar-scanner-6.2.0.4584-windows-x64/bin/sonar-scanner.bat ^
11         -D sonar.login=%SONAR_USER% ^
12         -D sonar.password=%SONAR_PASSWORD% ^
13         -D sonar.projectKey=exp-08 ^
14         -D sonar.exclusions=vendor/**,resources/**,**/*.java ^
15         -D sonar.host.url=http://127.0.0.1:9000/
16       """
17     }
18   }
19 }
  
```

If you would notice carefully, i have used the `withCredentials` function within the pipeline script which makes the build avail the facility of using the credentials that i had set earlier

I have also added a line where i have mentioned the path of the `sonar-scanner.bat` file, since there are problems with system configurations with the installation of soanrqube

Now, save and build the pipeline

Status: exp-08 (green)

made for exp 08

Permalinks:

- Last build (#6), 1 hr 16 min ago
- Last stable build (#6), 1 hr 16 min ago
- Last successful build (#6), 1 hr 16 min ago
- Last failed build (#5), 1 hr 20 min ago
- Last unsuccessful build (#5), 1 hr 20 min ago
- Last completed build (#6), 1 hr 16 min ago

Build History: trend

It generally takes 10-15 minutes for building this pipeline. We are supposed to stay patient and let the build process get completed

```

19:28:19.309 WARN Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/testelement/property/IntegerProperty.html for block at line 40. Keep
only the first 100 references.
19:28:19.309 WARN Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/testelement/property/IntegerProperty.html for block at line 41. Keep
only the first 100 references.
19:28:19.309 WARN Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/testelement/property/IntegerProperty.html for block at line 17. Keep
only the first 100 references.
19:28:19.309 WARN Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/testelement/property/IntegerProperty.html for block at line 849. Keep
only the first 100 references.
19:28:19.508 WARN Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/protocol/http/proxy/package-summary.html for block at line 39. Keep
only the first 100 references.
19:28:19.508 WARN Too many duplication references on file gameoflife-
web/tools/jmeter/docs/api/org/apache/jmeter/protocol/http/proxy/package-summary.html for block at line 40. Keep
only the first 100 references.

```

Dashboard > exp-08 > #6

```

19:20:20.114 INFO CRD EXECUTOR CRD CALCULATION FINISHING (idle) | LAMBDA@0.0.0.0:433c
19:28:28.428 INFO SCM revision ID 'ba799ba7e1b576f04a461232b0412c5e6e15e4'
19:30:36.377 INFO Analysis report generated in 8707ms, dir size=127.2 MB
19:30:48.802 INFO Analysis report compressed in 12408ms, zip size=29.6 MB
19:30:50.969 INFO Analysis report uploaded in 2154ms
19:30:50.970 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://127.0.0.1:9000/dashboard?id=exp-08
19:30:50.970 INFO Note that you will be able to access the updated dashboard once the server has processed the
submitted analysis report
19:30:50.970 INFO More about the report processing at http://127.0.0.1:9000/api/ce/task?id=a3814044-bd06-433c-
b530-2819edc58f24
19:31:08.715 INFO Analysis total time: 14:26.131 s
19:31:08.725 INFO SonarScanner Engine completed successfully
19:31:09.389 INFO EXECUTION SUCCESS
19:31:09.553 INFO Total time: 14:31.783s
[Pipeline]
[Pipeline] // withCredentials
[Pipeline]
[Pipeline] // withSonarQubeEnv
[Pipeline]
[Pipeline] // stage
[Pipeline]
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

After a successful build, we are supposed to go back to our Sonarqube profile. We will see this type of a message on the screen, where it offers us multiple options to analyze our code

sonarqube

Projects Issues Rules Quality Profiles Quality Gates Administration More Q

exp-08 main ?

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

Quality Gate Passed Last analysis 1 hour ago

New Code Overall Code

Security	Reliability	Maintainability
0 Open issues (A)	68k Open issues (C)	164k Open issues (A)
0 H 0 M 0 L	0 H 47k M 21k L	7 H 143k M 21k L

Within the issues section, let us explore different tabs

1. Bugs inside Type section

The screenshot shows the SonarQube interface for project 'exp-08'. The 'Issues' tab is selected. In the left sidebar under 'Type', 'Bug' is selected, showing 47k issues. A specific bug is highlighted with a blue border, labeled 'Add "lang" and/or "xml:lang" attributes to this "<html>" element'. It has an 'Intentionality' tag of 'Reliability' and 'accessibility wcag2-a' tags. The status is 'Open' and it was last updated 4 years ago.

2. Code Smell inside the Type section

The screenshot shows the SonarQube interface for project 'exp-08'. The 'Issues' tab is selected. In the left sidebar under 'Type', 'Code Smell' is selected, showing 164k issues. A specific code smell is highlighted with a blue border, labeled 'Use a specific version tag for the image.'. It has an 'Intentionality' tag of 'Maintainability' and 'No tags' tags. The status is 'Open' and it was last updated 4 years ago.

3. Consistency inside Clean Code Attribute

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration More A

exp-08 / main ✓ ?

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

Issues in new code

Clean Code Attribute 1 X

- Consistency 164k
- Intentionality 268
- Adaptability 0
- Responsibility 0

Add to selection Ctrl + click

Software Quality

Bulk Change Select issues ▲▼ Navigate to issue ←→ 163,766 issues 1705d effort

gameoflife-core/build/reports/tests/all-tests.html

Remove this deprecated "width" attribute. Consistency
Maintainability ● html5 obsolete +
Open Not assigned L9 · 5min effort · 4 years ago · Code Smell · Major

Remove this deprecated "align" attribute. Consistency
Maintainability ● html5 obsolete +

4. Reliability inside Software Quality

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration More A

exp-08 / main ✓ ?

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

Responsibility 0

Add to selection Ctrl + click

Software Quality 1 X

- Security 0
- Reliability** 21k
- Maintainability 164k

Add to selection Ctrl + click

Severity ?

Bulk Change Select issues ▲▼ Navigate to issue ←→ 20,856 issues 217d effort

gameoflife-core/build/reports/tests/all-tests.html

Anchors must have content and the content must be accessible by a screen reader. Consistency
Maintainability ● Reliability ● accessibility +
Open Not assigned L29 · 5min effort · 4 years ago · Code Smell · Minor

Anchors must have content and the content must be accessible by a screen reader. Consistency

5. Maintainability

The screenshot shows the SonarQube web interface for a project named 'exp-08'. The 'Issues' tab is selected. On the left, a sidebar displays quality profiles: Responsibility (0), Software Quality (1 issue), Security (0), Reliability (21k), and Maintainability (164k). The main panel shows a summary of 163,766 issues and 1705d effort. Below this, two specific issues are listed under the 'gameoflife-core/build/reports/tests/all-tests.html' file:

- Remove this deprecated "width" attribute.** (Consistency, Maintainability) - L9 • 5min effort • 4 years ago • Code Smell • Major
- Remove this deprecated "align" attribute.** (Consistency, Maintainability) - html5 obsolete

A note at the bottom states: "Embedded database should be used for evaluation purposes only".

Conclusion:

In this experiment, we set up a Jenkins CI/CD pipeline integrated with SonarQube to automate static analysis on a sample application. Jenkins was configured to trigger builds and run SonarQube's analysis with every code change, detecting bugs, code smells, and security vulnerabilities. This pipeline provided continuous monitoring and ensured early detection of issues, improving code quality and security. The experiment showcased how integrating CI/CD pipelines with SonarQube enhances development efficiency and ensures better, more reliable software.

Adv DevOps Exp 09

Aim: To Understand Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins and NRPE (Nagios Remote Plugin Executor) on Linux Machine.

Theory:

What is Nagios?

Nagios is an open-source software for continuous monitoring of systems, networks, and infrastructures. It runs plugins stored on a server that is connected with a host or another server on your network or the Internet. In case of any failure, Nagios alerts about the issues so that the technical team can perform the recovery process immediately.

Nagios is used for continuous monitoring of systems, applications, service and business processes in a DevOps culture.

Why We Need Nagios tool?

Here are the important reasons to use Nagios monitoring tool:

- Detects all types of network or server issues
- Helps you to find the root cause of the problem which allows you to get the permanent solution to the problem
- Active monitoring of your entire infrastructure and business processes
- Allows you to monitor and troubleshoot server performance issues
- Helps you to plan for infrastructure upgrades before outdated systems create failures
- You can maintain the security and availability of the service
- Automatically fix problems in a panic situation

Features of Nagios

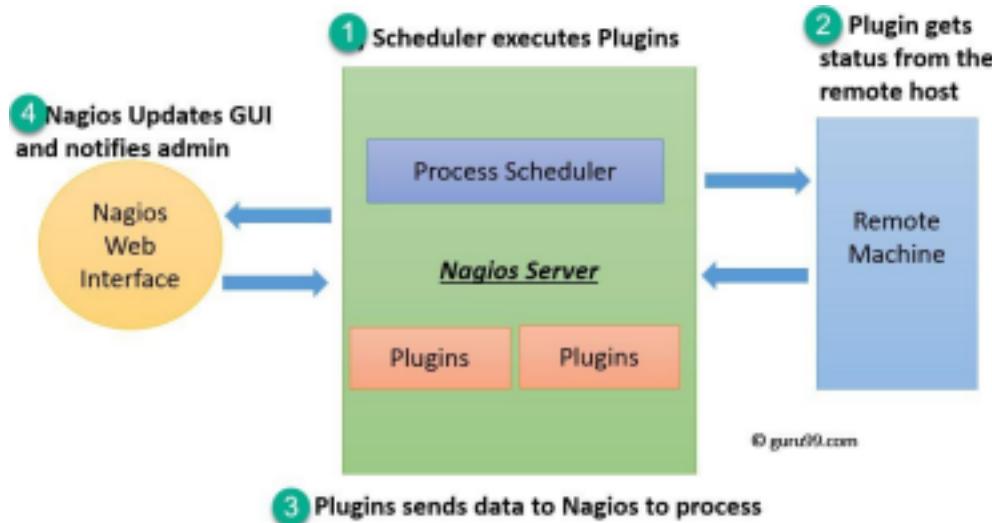
Following are the important features of Nagios monitoring tool:

- Relatively scalable, Manageable, and Secure
- Good log and database system
- Informative and attractive web interfaces
- Automatically send alerts if condition changes
- If the services are running fine, then there is no need to do check that host is an alive
- Helps you to detect network errors or server crashes
- You can troubleshoot the performance issues of the server.
- The issues, if any, can be fixed automatically as they are identified during the monitoring process
- You can monitor the entire business process and IT infrastructure with a single pass
- The product's architecture is easy to write new plugins in the language of your choice
- Nagios allows you to read its configuration from an entire directory which helps you to decide how to define individual files
- Utilizes topology to determine dependencies
- Monitor network services like HTTP, SMTP, HTTP, SNMP, FTP, SSH, POP, etc.
- Helps you to define network host hierarchy using parent hosts
- Ability to define event handlers that runs during service or host events for proactive

- problem resolution
- Support for implementing redundant monitoring hosts

Nagios Architecture

Nagios is a client-server architecture. Usually, on a network, a Nagios server is running on a host, and plugins are running on all the remote hosts which should be monitored.



1. The scheduler is a component of the server part of Nagios. It sends a signal to execute the plugins at the remote host.
2. The plugin gets the status from the remote host
3. The plugin sends the data to the process scheduler
4. The process scheduler updates the GUI and notifications are sent to admins.

Step 1: Create a security group with the required configurations

I have created a new security group with a name 'newsecurity'

EC2 > Security Groups > Create security group

Create security group Info

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, you must provide a name and optional description. You can also specify the VPC that the security group belongs to.

Basic details

Security group name Info

Name cannot be edited after creation.

Description Info

VPC Info

I have modified the INBOUND RULES as follows

The screenshot shows the AWS Management Console interface for managing security group inbound rules. The top navigation bar includes the AWS logo, Services, a search bar, and account information. The main section is titled "Inbound rules Info". It displays a table with columns for Type, Protocol, Port range, Source, and Description - optional. There are seven rows of rules:

Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	Anywhere	:/0 X
HTTPS	TCP	443	Anywhere	0.0.0.0/0 X
SSH	TCP	22	Anywhere	0.0.0.0/0 X
All ICMP - IPv6	IPv6 ICMP	All	Anywhere	:/0 X
All ICMP - IPv4	ICMP	All	Anywhere	0.0.0.0/0 X
All traffic	All	All	Anywhere	0.0.0.0/0 X
Custom TCP	TCP	5666	Anywhere	0.0.0.0/0 X

Step 2: Create ec2 instance

Name it as nagios-host. Select instance type as amazon-linux and choose the already created key pair and security group

The screenshot shows the AWS Lambda console interface. A modal window is open for creating a new function. The 'Function name' field contains 'HelloWorld'. Under 'Runtime', 'Node.js 12.x' is selected. The 'Handler' field shows 'index.handler'. In the 'Code' section, 'Upload a ZIP file' is selected, and a file named 'lambda_function.zip' is uploaded. The 'Configure' tab is selected. On the left sidebar, 'HelloWorld' is listed under 'My Functions'.

The screenshot shows the AWS Lambda console interface. A modal window is open for configuring the 'HelloWorld' function. Under 'Basic settings', the 'Function name' is 'HelloWorld', 'Runtime' is 'Node.js 12.x', and 'Handler' is 'index.handler'. The 'Code' section shows 'Upload a ZIP file' selected, with 'lambda_function.zip' uploaded. The 'Configure' tab is selected. On the left sidebar, 'HelloWorld' is listed under 'My Functions'.

The screenshot shows the AWS Lambda console interface. A modal window is open for configuring the 'HelloWorld' function. Under 'Basic settings', the 'Function name' is 'HelloWorld', 'Runtime' is 'Node.js 12.x', and 'Handler' is 'index.handler'. The 'Code' section shows 'Upload a ZIP file' selected, with 'lambda_function.zip' uploaded. The 'Configure' tab is selected. On the left sidebar, 'HelloWorld' is listed under 'My Functions'.

Copy the given ssh command, as we will require it for logging into our nagios-host instance from our windows powershell

EC2 > Instances > i-0eda234d2e9ec8648 > Connect to instance

Connect to instance Info

Connect to your instance i-0eda234d2e9ec8648 (nagios-host) using any of these options

EC2 Instance Connect | Session Manager | **SSH client** | EC2 serial console

Instance ID
i-0eda234d2e9ec8648 (nagios-host)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is mohit.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
 chmod 400 "mohit.pem"
4. Connect to your instance using its Public DNS:
 ec2-54-81-152-209.compute-1.amazonaws.com

Command copied

ssh -i "mohit.pem" ec2-user@ec2-54-81-152-209.compute-1.amazonaws.com

Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Cancel

Step 3: Open an administrative powershell and remotely login using the above mentioned ssh command

```
ec2-user@ip-172-31-92-249:~ 
Windows PowerShell
Copyright (c) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> cd C:\Users\Del1\Downloads
PS C:\Users\Del1\Downloads> ssh -i "mohit.pem" ec2-user@ec2-54-81-152-209.compute-1.amazonaws.com
, # ~\_ ##### Amazon Linux 2023
~~ \#####\ ~\##| ~~ \#/ https://aws.amazon.com/linux/amazon-linux-2023
~~ \#/\ \-\> ~~ V~'`-\>
~~ / ~~ / 
~~ / \ / 
~~ / \ / 
Last login: Mon Sep 30 09:25:13 2024 from 125.99.93.18
, # ~\_ ##### Amazon Linux 2023
~~ \#####\ ~\##| ~~ \#/ https://aws.amazon.com/linux/amazon-linux-2023
~~ \#/\ \-\> ~~ V~'`-\>
~~ / ~~ / 
~~ / \ / 
Last login: Mon Sep 30 09:25:13 2024 from 125.99.93.18
[ec2-user@ip-172-31-92-249 ~]$ sudo yum update
Last metadata expiration check: 0:13:13 ago on Mon Sep 30 09:23:03 2024.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-92-249 ~]$ sudo yum install httpd php
Last metadata expiration check: 0:13:23 ago on Mon Sep 30 09:23:03 2024.
Package httpd-2.4.62-1.amzn2023.x86_64 is already installed.
Package php8.3-8.3.10-1.amzn2023.0.1.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

And then run these commands
sudo yum update
sudo yum install httpd php

```
[ec2-user@ip-172-31-41-160 ~]$ sudo yum update
Last metadata expiration check: 0:01:37 ago on Wed Oct 2 12:28:33 2024.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-41-160 ~]$ sudo yum install httpd php
Last metadata expiration check: 0:01:45 ago on Wed Oct 2 12:28:33 2024.
Dependencies resolved.

=====
Package           Architecture      Version       Repository   Size
=====
Installing:
httpd            x86_64          2.4.62-1.amzn2023
php8_3           x86_64          8.3.10-1.amzn2023.0.1

Installing dependencies:
apr              x86_64          1.7.2-2.amzn2023.0.2
apr-util         x86_64          1.6.3-1.amzn2023.0.1
generic-logs-httd noarch        18.0.0-12.amzn2023.0.3
httpd-core       x86_64          2.4.62-1.amzn2023
httpd-filesystem noarch        2.4.62-1.amzn2023
httpd-tools      x86_64          2.4.62-1.amzn2023
libcurl          x86_64          1.0.19-4.amzn2023.0.2
libcurl-ssl     x86_64          1.0.19-5.amzn2023
libcurl-xml     x86_64          1.0.19-5.amzn2023
libcurl-zlib    x86_64          1.0.19-5.amzn2023.0.2
libxml2          noarch        2.1.49-3.amzn2023.0.3
mailing          noarch        1.1.24-0.1.amzn2023.0.4
nginx-filesystem noarch        2.0.1-1.amzn2023.0.1
nginx          x86_64          2.0.1-1.amzn2023.0.1
php8_3-common   x86_64          8.3.10-1.amzn2023.0.1
php8_3-process  x86_64          8.3.10-1.amzn2023.0.1
php8_3-xml     x86_64          8.3.10-1.amzn2023.0.1

Installing weak dependencies:
apr-util-openssl x86_64          1.6.3-1.amzn2023.0.1
mod_ssl          x86_64          2.5.22-1.amzn2023.0.3
mod_wsgi         x86_64          2.4.62-1.amzn2023
php8_3-ftp      x86_64          8.3.10-1.amzn2023.0.1
php8_3-mbstring x86_64          8.3.10-1.amzn2023.0.1
php8_3-opcache  x86_64          8.3.10-1.amzn2023.0.1
php8_3-pdo     x86_64          8.3.10-1.amzn2023.0.1
php8_3-sodium   x86_64          8.3.10-1.amzn2023.0.1

Transaction Summary
Install 25 Packages
```

sudo yum install gcc glibc glibc-common

```
[ec2-user@ip-172-31-41-160 ~]$ sudo yum install gcc glibc glibc-common
Last metadata expiration check: 0:02:02 ago on Wed Oct 2 12:28:33 2024.
Package glibc-2.34-52.amzn2023.0.11.x86_64 is already installed.
Package glibc-common-2.34-52.amzn2023.0.11.x86_64 is already installed.
Dependencies resolved.

=====
Package           Architecture      Version       Repository   Size
=====
Installing:
gcc              x86_64          11.4.1-2.amzn2023.0.2
Installing dependencies:
annobin-docs      noarch        10.93-1.amzn2023.0.1
annobin-plugin-gcc x86_64          10.93-1.amzn2023.0.1
cpp              x86_64          11.4.1-2.amzn2023.0.2
gc               x86_64          8.0.4-5.amzn2023.0.2
glibc-devel      x86_64          2.34-52.amzn2023.0.11
glibc-headers-x86 noarch        2.34-52.amzn2023.0.11
guile22         x86_64          2.2.7-2.amzn2023.0.3
kernel-headers   x86_64          6.1.109-118.189.amzn2023
libmpc          x86_64          1.2.1-2.amzn2023.0.2
libtool-ltdl    x86_64          2.4.7-1.amzn2023.0.3
libcrypt-devel  x86_64          4.4.33-7.amzn2023
make             x86_64          1:4.3-5.amzn2023.0.2

Transaction Summary
Install 13 Packages

Total download size: 52 M
Installed size: 168 M
Is this ok [y/N]: y
Downloading Packages:
(1/13): annobin-docs-10.93-1.amzn2023.0.1.noarch.rpm                                852 kB/s |  92 kB  00:00
(2/13): annobin-plugin-gcc-10.93-1.amzn2023.0.1.x86_64.rpm                          6.5 MB/s | 887 kB  00:00
(3/13): gc-8.0.4-5.amzn2023.0.2.x86_64.rpm                                         2.3 MB/s | 105 kB  00:00
(4/13): glibc-devel-2.34-52.amzn2023.0.11.x86_64.rpm                           1.1 MB/s | 27 kB  00:00
(5/13): cpp-11.4.1-2.amzn2023.0.2.x86_64.rpm                                     32 MB/s | 10 MB  00:00
(6/13): glibc-headers-x86-2.34-52.amzn2023.0.11.noarch.rpm                      2.9 MB/s | 427 kB  00:00
(7/13): kernel-headers-6.1.109-118.189.amzn2023.x86_64.rpm                         16 MB/s | 1.4 MB  00:00
(8/13): libmpc-1.2.1-2.amzn2023.0.2.x86_64.rpm                               2.1 MB/s | 62 kB  00:00
(9/13): guile22-2.2.7-2.amzn2023.0.3.x86_64.rpm                            27 MB/s | 6.4 MB  00:00
(10/13): libtool-ltdl-2.4.7-1.amzn2023.0.3.x86_64.rpm                         322 kB/s | 38 kB  00:00
(11/13): libcrypt-devel-4.4.33-7.amzn2023.x86_64.rpm                         1.4 MB/s | 32 kB  00:00
```

sudo yum install gd gd-devel

```
[ec2-user@ip-172-31-41-160 ~]$ sudo yum install gd gd-devel
Last metadata expiration check: 0:02:25 ago on Wed Oct 2 12:28:33 2024.
Dependencies resolved.

=====
Package           Architecture Version      Repository  Size
=====
Installing:
gd               x86_64     2.3.3-5.amzn2023.0.3   amazonlinux 139 k
gd-devel         x86_64     2.3.3-5.amzn2023.0.3   amazonlinux 38 k
Installing dependencies:
brotli           x86_64     1.0.9-4.amzn2023.0.2   amazonlinux 314 k
brotli-devel     x86_64     1.0.9-4.amzn2023.0.2   amazonlinux 31 k
bzip2-devel      x86_64     1.0.8-6.amzn2023.0.2   amazonlinux 214 k
cairo             x86_64     1.17.6-2.amzn2023.0.1   amazonlinux 684 k
cmake-filesystem x86_64     3.22.2-1.amzn2023.0.4   amazonlinux 16 k
fontconfig        x86_64     2.13.94-2.amzn2023.0.2   amazonlinux 273 k
fontconfig-devel x86_64     2.13.94-2.amzn2023.0.2   amazonlinux 128 k
fonts-filesystem noarch    1:2.0.5-12.amzn2023.0.2   amazonlinux 9.5 k
freetype          x86_64     2.13.2-5.amzn2023.0.1   amazonlinux 423 k
freetype-devel   x86_64     2.13.2-5.amzn2023.0.1   amazonlinux 912 k
glib2-devel      x86_64     2.74.7-689.amzn2023.0.2   amazonlinux 486 k
google-noto-fonts-common x86_64   29201286-1.amzn2023.0.2   amazonlinux 15 k
google-noto-sans-vf-fonts x86_64   29201286-1.amzn2023.0.2   amazonlinux 492 k
graphite2         x86_64     1.3.14-7.amzn2023.0.2   amazonlinux 97 k
graphite2-devel  x86_64     1.3.14-7.amzn2023.0.2   amazonlinux 21 k
harfbuzz          x86_64     7.0.0-2.amzn2023.0.1   amazonlinux 868 k
harfbuzz-devel   x86_64     7.0.0-2.amzn2023.0.1   amazonlinux 404 k
harfbuzz-icu     x86_64     7.0.0-2.amzn2023.0.1   amazonlinux 18 k
jbigkit-libs      x86_64     2.1.21.amzn2023.0.2   amazonlinux 54 k
langpacks-core-font-en noarch   3.0-21.amzn2023.0.4   amazonlinux 10 k
libICE            x86_64     1.0.10-6.amzn2023.0.2   amazonlinux 71 k
libSM             x86_64     1.2.3-8.amzn2023.0.2   amazonlinux 42 k
libX11            x86_64     1.7.2-2.amzn2023.0.4   amazonlinux 657 k
libX11-common    x86_64     1.7.2-3.amzn2023.0.4   amazonlinux 152 k
libX11-devel     x86_64     1.7.2-3.amzn2023.0.4   amazonlinux 939 k
libX11-xcb      x86_64     1.7.2-3.amzn2023.0.4   amazonlinux 12 k
libXau            x86_64     1.0.9-6.amzn2023.0.2   amazonlinux 31 k
libXau-devel     x86_64     1.0.9-6.amzn2023.0.2   amazonlinux 14 k
libXext           x86_64     1.3.4-6.amzn2023.0.2   amazonlinux 41 k
libXpm            x86_64     3.5.15-2.amzn2023.0.3   amazonlinux 65 k
libXpm-devel     x86_64     3.5.15-2.amzn2023.0.3   amazonlinux 59 k
libXrender        x86_64     0.9.10-14.amzn2023.0.2   amazonlinux 28 k
libXt             x86_64     1.2.0-4.amzn2023.0.2   amazonlinux 181 k
libblkid-devel   x86_64     2.37.4-1.amzn2023.0.4   amazonlinux 15 k
```

Create a new Nagios User with its password. You'll have to enter the password twice for confirmation.

sudo adduser -m nagios
sudo passwd nagios

```
[ec2-user@ip-172-31-41-160 ~]$ sudo adduser -m nagios
[ec2-user@ip-172-31-41-160 ~]$ sudo passwd nagios
Changing password for user nagios.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-41-160 ~]$
```

Create a new user group & create a new directory for Nagios downloads using the following commands

sudo groupadd nagcmd
sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache
mkdir ~/downloads
cd ~/downloads

Use **wget** to download the source zip files.

In this step, we are downloading, the latest version of nagios and the necessary plugins required to carry out the tasks of setting up a nagios server

wget <https://sourceforge.net/projects/nagios/files/latest/download>

```
[ec2-user@ip-172-31-41-160:~/downloads/nagios-4.5.5]
[ec2-user@ip-172-31-41-160 downloads]$ wget https://sourceforge.net/projects/nagios/files/latest/download
--2024-10-02 12:34:21-- https://sourceforge.net/projects/nagios/files/latest/download
Resolving sourceforge.net (sourceforge.net)... 172.64.150.145, 104.18.37.111, 2606:4700:4400::6812:256f, ...
Connecting to sourceforge.net (sourceforge.net)|172.64.150.145|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://downloads.sourceforge.net/project/nagios/nagios-4.x/nagios-4.5.5/nagios-4.5.5.tar.gz?ts=gAAAAABm_T3NmNSZPzP-6la2Tltvo0GCG7VVV7QGVH08n3tC240ehfMw7vhCoKbGHg2iIRxbmfugI10LccNfxtaoixg3jzKg3w3D0use_mirror=phoenixnapkr=[following]
--2024-10-02 12:34:21-- https://downloads.sourceforge.net/project/nagios/nagios-4.x/nagios-4.5.5/nagios-4.5.5.tar.gz?ts=gAAAAABm_T3NmNSZPzP-6la2Tltvo0GCG7VVV7QGVH08n3tC240ehfMw7vhCoKbGHg2iIRxbmfugI10LccNfxtaoixg3jzKg3w3D0use_mirror=phoenixnapkr=
Resolving downloads.sourceforge.net (downloads.sourceforge.net)|204.68.111.105|:443... connected.
Connecting to downloads.sourceforge.net (downloads.sourceforge.net)|204.68.111.105|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2065473 (2.0M) [application/x-gzip]
Saving to: 'download'

download                                         100%[=====] 1.97M 4.23MB/s in 0.5s

2024-10-02 12:34:22 (4.23 MB/s) - 'download' saved [2065473/2065473]
```

wget <https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz>

```
[ec2-user@ip-172-31-41-160:~/downloads/nagios-4.5.5]
[ec2-user@ip-172-31-41-160 downloads]$ wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
--2024-10-02 12:34:46-- https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
Resolving nagios-plugins.org (nagios-plugins.org)... 45.56.123.251
Connecting to nagios-plugins.org (nagios-plugins.org)|45.56.123.251|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2753049 (2.0M) [application/x-gzip]
Saving to: 'nagios-plugins-2.4.11.tar.gz'

nagios-plugins-2.4.11.tar.gz 100%[=====] 2.62M 7.48MB/s in 0.4s

2024-10-02 12:34:46 (7.48 MB/s) - 'nagios-plugins-2.4.11.tar.gz' saved [2753049/2753049]
```

```
[ec2-user@ip-172-31-92-249:~/downloads]
[ec2-user@ip-172-31-92-249 ~]$ cd ~/downloads
[ec2-user@ip-172-31-92-249 downloads]$ wget https://sourceforge.net/projects/nagios/files/latest/download
--2024-09-30 09:54:56-- https://sourceforge.net/projects/nagios/files/latest/download
Resolving sourceforge.net (sourceforge.net)... 172.64.150.145, 104.18.37.111, 2606:4700:4400::6812:256f, ...
Connecting to sourceforge.net (sourceforge.net)|172.64.150.145|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://downloads.sourceforge.net/project/nagios/nagios-4.x/nagios-4.5.5/nagios-4.5.5.tar.gz?ts=gAAAAABm-nVw9RdAvnMaShLf3gu4RXTSVxrTZ6fGxJvhVAOzpB1bPgbyzLMcDDAALgtEC1p0Kr0cgJNJ23bKktan1cJ0Vfkpg3D0use_mirror=netaactuate&r=[following]
--2024-09-30 09:54:56-- https://downloads.sourceforge.net/project/nagios/nagios-4.x/nagios-4.5.5/nagios-4.5.5.tar.gz?ts=gAAAAABm-nVw9RdAvnMaShLf3gu4RXTSVxrTZ6fGxJvhVAOzpB1bPgbyzLMcDDAALgtECl0OKrcpgJNj23bKktan1cJ0Vfkpg3D0use_mirror=netaactuate&r=
Resolving downloads.sourceforge.net (downloads.sourceforge.net)|204.68.111.105|:443... connected.
Connecting to downloads.sourceforge.net (downloads.sourceforge.net)|204.68.111.105|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2065473 (2.0M) [application/x-gzip]
Saving to: 'download'

download                                         100%[=====] 1.97M ---KB/s in 0.07s

2024-09-30 09:54:57 (29.8 MB/s) - 'download' saved [2065473/2065473]

[ec2-user@ip-172-31-92-249 downloads]$ wget https://nagios-plugins.org/download/nagios-plugins-2.4.9.tar.gz
--2024-09-30 09:56:53-- https://nagios-plugins.org/download/nagios-plugins-2.4.9.tar.gz
Resolving nagios-plugins.org (nagios-plugins.org)... 45.56.123.251
Connecting to nagios-plugins.org (nagios-plugins.org)|45.56.123.251|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2754403 (2.0M) [application/x-gzip]
Saving to: 'nagios-plugins-2.4.9.tar.gz'

nagios-plugins-2.4.9.tar.gz 100%[=====] 2.63M 7.54MB/s in 0.3s

2024-09-30 09:56:54 (7.54 MB/s) - 'nagios-plugins-2.4.9.tar.gz' saved [2754403/2754403]
```

Now, we run the next command in the following manner

tar zxvf <nagios-4.5.5 version> (for me it has gotten saved as 'download')

So i wrote tar zxvf download

```
[ec2-user@ip-172-31-41-160:~/downloads]
[nagios-4.5.5]
[nagios-4.5.5/.github]
[nagios-4.5.5/.github/workflows]
[nagios-4.5.5/.github/workflows/test.yml]
[nagios-4.5.5/.gitignore]
[nagios-4.5.5/.CONTRIBUTING.md]
[nagios-4.5.5/Changelog]
[nagios-4.5.5/INSTALLING]
[nagios-4.5.5/LEGAL]
[nagios-4.5.5/LICENSE]
[nagios-4.5.5/Makefile.in]
[nagios-4.5.5/README.md]
[nagios-4.5.5/THANKS]
[nagios-4.5.5/UPGRADING]
[nagios-4.5.5/autocal.m4]
[nagios-4.5.5/autocom-macros]
[nagios-4.5.5/autocom-macros/.gitignore]
[nagios-4.5.5/autocom-macros/CHANGELOG.md]
[nagios-4.5.5/autocom-macros/LICENSE]
[nagios-4.5.5/autocom-macros/LICENSE.md]
[nagios-4.5.5/autocom-macros/README.md]
[nagios-4.5.5/autocom-macros/add_group_user]
[nagios-4.5.5/autocom-macros/ax_nagios_get_distrib]
[nagios-4.5.5/autocom-macros/ax_nagios_get_files]
[nagios-4.5.5/autocom-macros/ax_nagios_get_inetd]
[nagios-4.5.5/autocom-macros/ax_nagios_get_init]
[nagios-4.5.5/autocom-macros/ax_nagios_get_os]
[nagios-4.5.5/autocom-macros/ax_nagios_get_paths]
[nagios-4.5.5/autocom-macros/ax_nagios_get_ssl]
[nagios-4.5.5/base]
[nagios-4.5.5/base/.gitignore]
[nagios-4.5.5/base/Makefile.in]
[nagios-4.5.5/base/broker.c]
[nagios-4.5.5/base/checks.c]
[nagios-4.5.5/base/commands.c]
[nagios-4.5.5/base/config.c]
[nagios-4.5.5/base/events.c]
[nagios-4.5.5/base/flapping.c]
[nagios-4.5.5/base/logging.c]
[nagios-4.5.5/base/nagios.c]
[nagios-4.5.5/base/nagiosstats.c]
[nagios-4.5.5/base/nemodus.c]
[nagios-4.5.5/base/nerc.c]
```

After which we are supposed to **change our directory** over there

For eg. **cd nagios-4.5.5...** depending on the version that we have downloaded

Next, Run this command (make sure that you are working inside nagios-4.x.x directory)

./configure --with-command-group=nagcmd

```
[ec2-user@ip-172-31-41-160:~/downloads/nagios-4.5.5]
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ ./configure --with-command-group=nagcmd
checking for a BSD-compatible install... /usr/bin/install -c
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking for suffix of object files... o
checking whether the compiler supports GNU C... yes
checking whether gcc accepts -g... yes
checking for gcc option to enable C11 features... none needed
checking whether make sets ${MAKE}... yes
checking whether ln -s works... yes
checking for strip... /usr/bin/strip
checking for sys/wait.h that is POSIX.1 compatible... yes
checking for stdio.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for inttypes.h... yes
checking for stdint.h... yes
checking for strings.h... yes
checking for sys/stat.h... yes
checking for sys/types.h... yes
checking for unistd.h... yes
checking for arpa/inet.h... yes
checking for ctype.h... yes
checking for dirent.h... yes
checking for errno.h... yes
checking for fcntl.h... yes
checking for getopt.h... yes
checking for grp.h... yes
checking for libgen.h... yes
checking for limits.h... yes
checking for math.h... yes
checking for netdb.h... yes
checking for netinet/in.h... yes
checking for pwd.h... yes
checking for regex.h... yes
checking for signal.h... yes
checking for socket.h... no
checking for stdarg.h... yes
```

```
checking for server... yes
checking for strtoul... yes
checking for unsetenv... yes
checking for type of socket size... size_t
checking for Kerberos include files... configure: WARNING: could not find include files
checking for pkg-config... pkg-config
checking for SSL headers... configure: error: Cannot find ssl headers
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ ■
```

After running this command, we get an **error related to ssl header being absent**

For that purpose, we are to run the following command.

sudo yum install openssl-devel (for ssl header)

```
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo yum install openssl-devel
Last metadata expiration check: 0:02:11 ago on Wed Oct 2 12:28:33 2024.
Dependencies resolved.
=====
Transaction Summary
=====
Install 1 Package

Total download size: 3.0 M
Installed size: 4.7 M
Is this ok [y/N]: y
Downloading Packages:
openssl-devel-3.0.8-1.amzn2023.0.14.x86_64.rpm

Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing :
  Installing : openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64
  Running scriptlet: openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64
  Verifying : openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64

Installed:
  openssl-devel-1:3.0.8-1.amzn2023.0.14.x86_64

Complete!
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ ■
```

Now, Re-run **./configure --with-command-group=nagcmd**

After this, run **make all** command

```
[ec2-user@ip-172-31-41-160:~/downloads/nagios-4.5.5$ make all
cd ./base && make
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o nagios.o ./nagios.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o broker.o broker.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o nebmods.o nebmods.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o .../common/shared.o .../common/shared.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o query-handler.o query-handler.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o workers.o workers.c
In function `get_wproc_list',
  inlined from `get_worker' at workers.c:277:12:
workers.c:253:17: warning: %s` directive argument is null [-Wformat=overflow]
  253 |     log_debug_info(DEBUGL_CHECKS, 1, "found specialized worker(s) for \"%s\", (%s&&*%s!= '/') ? slash : cmd_name");
  |             ~~~~~
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o checks.o checks.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o config.o config.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o commands.o commands.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o events.o events.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o flapping.o flapping.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o logger.o logger.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o macros.o macros.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o netutils.o netutils.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o notifications.o notifications.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o seahandlers.o seahandlers.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o utils.o utils.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o retention-base.o .../retention.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o xretention-base.o .../xdata/xrdddefault.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o comments-base.o .../common/comments.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o xcomments-base.o .../xdata/xcddefaul.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o objects-base.o .../common/objects.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o xobjects-base.o .../xdata/xodtemplate.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o statustatusdata.o .../common/statustdata.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o xstatusdata-base.o .../xdata/xsdddefaut.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o perfdata-base.o .../perfdata.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o xperfdata-base.o .../xdata/xpddefaul.c
gcc -Wall -I.. -I.. -I./lib -I./include -I./include -I./include -g -O2 -DHAVE_CONFIG_H -DSCORE -c -o downtime-base.o .../common/downtime.c
make C ..lib
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/lib'
gcc -Wall -g -O2 -I.. -I./include -DHAVE_CONFIG_H -c squeue.c -o squeue.o
gcc -Wall -g -O2 -I.. -I./include -DHAVE_CONFIG_H -c kvvec.c -o kvvec.o
gcc -Wall -g -O2 -I.. -I./include -DHAVE_CONFIG_H -c iocache.c -o iocache.o
gcc -Wall -g -O2 -I.. -I./include -DHAVE_CONFIG_H -c bitmap.o -o bitmap.o
gcc -Wall -g -O2 -I.. -I./include -DHAVE_CONFIG_H -c dkhash.o -o dkhash.o
```

```

ec2-user@ip-172-31-41-160:~/downloads/nagios-4.5.5
on doing this. Pay particular attention to the docs on
object configuration files, as they determine what/how
things get monitored!
make install-webconf
  This installs the Apache config file for the Nagios
  web interface
make install-exfoliation
  This installs the Exfoliation theme for the Nagios
  web interface
make install-classicui
  This installs the classic theme for the Nagios
  web interface

*** Support Notes *****
If you have questions about configuring or running Nagios,
please make sure that you:
  - Look at the sample config files
  - Read the documentation on the Nagios Library at:
    https://library.nagios.com

before you post a question to one of the mailing lists.
Also make sure to include pertinent information that could
help others help you. This might include:
  - What version of Nagios you are using
  - What version of the plugins you are using
  - Relevant snippets from your config files
  - Relevant error messages from the Nagios log file

For more information on obtaining support for Nagios, visit:
  https://support.nagios.com

*****
Enjoy.

```

Run the following set of commands to ensure that
sudo make install

```

ec2-user@ip-172-31-41-160:~/downloads/nagios-4.5.5
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo make install
cd ./base && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagiostats /usr/local/nagios/bin
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/base'
cd ./cgi && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make install-basic
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/sbin
for file in *.cgi; do \
    /usr/bin/install -c -s -m 775 -o nagios -g nagios $file /usr/local/nagios/sbin; \
done
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
cd ./html && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/html'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/media
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/stylesheets
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/contexthelp
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs/images
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/js
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/images
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/images/logos
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/includes
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/ssi
/usr/bin/install -c -m 664 -o nagios -g nagios ./robots.txt /usr/local/nagios/share
/usr/bin/install -c -m 664 -o nagios -g nagios ./jsonquery.html /usr/local/nagios/share
rm -f /usr/local/nagios/share/index.html
rm -f /usr/local/nagios/share/main.html
rm -f /usr/local/nagios/share/side.html
rm -f /usr/local/nagios/share/map.html
rm -f /usr/local/nagios/share/rss/*
rm -f /usr/local/nagios/share/graph-header.html
rm -f /usr/local/nagios/share/histogram.html
rm -f /usr/local/nagios/share/histogram-form.html
rm -f /usr/local/nagios/share/histogram-graph.html
rm -f /usr/local/nagios/share/histogram-links.html
rm -f /usr/local/nagios/share/infobox.html
rm -f /usr/local/nagios/share/map.php

```

sudo make install-init

```
[ec2-user@ip-172-31-41-160:~/downloads/nagios-4.5.5]
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo make install-init
/usr/bin/install -c -m 755 -d -o root -g root /lib/systemd/system
/usr/bin/install -c -m 755 -o root -g root startup/default-service /lib/systemd/system/nagios.service
```

sudo make install-config

```
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo make install-config
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc/objects
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/nagios.cfg /usr/local/nagios/etc/nagios.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/cgi.cgi /usr/local/nagios/etc/cgi.cgi
/usr/bin/install -c -b -m 660 -o nagios -g nagios sample-config/resource.cfg /usr/local/nagios/etc/resource.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/templates.cfg /usr/local/nagios/etc/objects/templates.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/commands.cfg /usr/local/nagios/etc/objects/commands.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/contacts.cfg /usr/local/nagios/etc/objects/contacts.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/timeperiods.cfg /usr/local/nagios/etc/objects/timeperiods.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/localhost.cfg /usr/local/nagios/etc/objects/localhost.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/windows.cfg /usr/local/nagios/etc/objects/windows.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/printer.cfg /usr/local/nagios/etc/objects/printer.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/switch.cfg /usr/local/nagios/etc/objects/switch.cfg

*** Config files installed ***

Remember, these are *SAMPLE* config files. You'll need to read
the documentation for more information on how to actually define
services, hosts, etc. to fit your particular needs.
```

sudo make install-webconf

```
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagios.conf
if [ 0 -eq 1 ]; then \
    ln -s /etc/httpd/conf.d/nagios.conf /etc/apache2/sites-enabled/nagios.conf; \
fi

*** Nagios/Apache conf file installed ***

[ec2-user@ip-172-31-41-160 nagios-4.5.5]$
```

Next, we are supposed to create a nagiosadmin account for nagios login along with password.
Specify the password twice.

sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin

```
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$
```

Restart Apache

sudo service httpd restart

Go back to the downloads folder and unzip the plugins zip file.

cd ~/downloads

tar zxvf nagios-plugins-2.4.11.tar.gz

```
[ec2-user@ip-172-31-41-160:~/downloads]
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[ec2-user@ip-172-31-41-160 nagios-4.5.5]$ cd ~/downloads
[ec2-user@ip-172-31-41-160 downloads]$ tar zxvf nagios-plugins-2.4.11.tar.gz
nagios-plugins-2.4.11/
nagios-plugins-2.4.11/build-aux/
nagios-plugins-2.4.11/build-aux/compile
nagios-plugins-2.4.11/build-aux/config.guess
nagios-plugins-2.4.11/build-aux/config.rpath
nagios-plugins-2.4.11/build-aux/config.sub
nagios-plugins-2.4.11/build-aux/install-sh
nagios-plugins-2.4.11/build-aux/ltdmain.sh
nagios-plugins-2.4.11/build-aux/missing
nagios-plugins-2.4.11/build-aux/mkinstalldirs
nagios-plugins-2.4.11/build-aux/depcomp
nagios-plugins-2.4.11/build-aux/snippet/
nagios-plugins-2.4.11/build-aux/snippet/_Noreturn.h
nagios-plugins-2.4.11/build-aux/snippet/arg-nonnull.h
nagios-plugins-2.4.11/build-aux/snippet/c++defs.h
nagios-plugins-2.4.11/build-aux/snippet/warn-on-use.h
nagios-plugins-2.4.11/build-aux/test-driver
```

Compile and install plugins

cd nagios-plugins-2.4.11

./configure --with-nagios-user=nagios --with-nagios-group=nagios

Run the following command:

sudo chkconfig --add nagios

On running the above command

```
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo chkconfig --add nagios
error reading information on service nagios: No such file or directory
```

If this is the output that one is getting, then it means that the init script is missing...

We can check this by running ls /etc/init.d/

```
[ec2-user@ip-172-31-92-249 nagios-plugins-2.4.9]$ ls /etc/init.d/
README  functions
[ec2-user@ip-172-31-92-249 nagios-plugins-2.4.9]$
```

With ls command, we must see a file named nagios, which i was not able to see

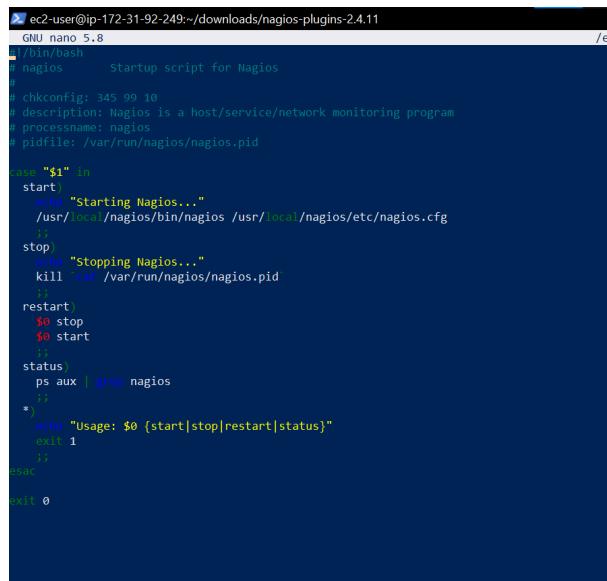
If the Init Script is Missing i.e If you don't see the nagios script in /etc/init.d/, you can create it manually. Here's how:

Run the following command:

sudo nano /etc/init.d/nagios

Within this file, paste the following script

```
#!/bin/bash
# nagios      Startup script for Nagios
#
# chkconfig: 345 99 10
# description: Nagios is a host/service/network monitoring program
# processname: nagios
# pidfile: /var/run/nagios/nagios.pid
case "$1" in
    start)
        echo "Starting Nagios..."
        /usr/local/nagios/bin/nagios /usr/local/nagios/etc/nagios.cfg
        ;;
    stop)
        echo "Stopping Nagios..."
        kill `cat /var/run/nagios/nagios.pid`
        ;;
    restart)
        $0 stop
        $0 start
        ;;
    status)
        ps aux | grep nagios
        ;;
    *)
        echo "Usage: $0 {start|stop|restart|status}"
        exit 1
        ;;
esac
exit 0
```



The screenshot shows a terminal window titled 'ec2-user@ip-172-31-92-249:~/downloads/nagios-plugins-2.4.11'. The window displays the script content from the previous code block. The text is in white on a dark background, with syntax highlighting for commands like '#!', '\$1', and file paths.

Make the Script Executable: After saving the file, run the following command to make it executable:

```
sudo chmod +x /etc/init.d/nagios
```

Run `sudo chkconfig --add nagios` again

And then run `sudo chkconfig nagios on`

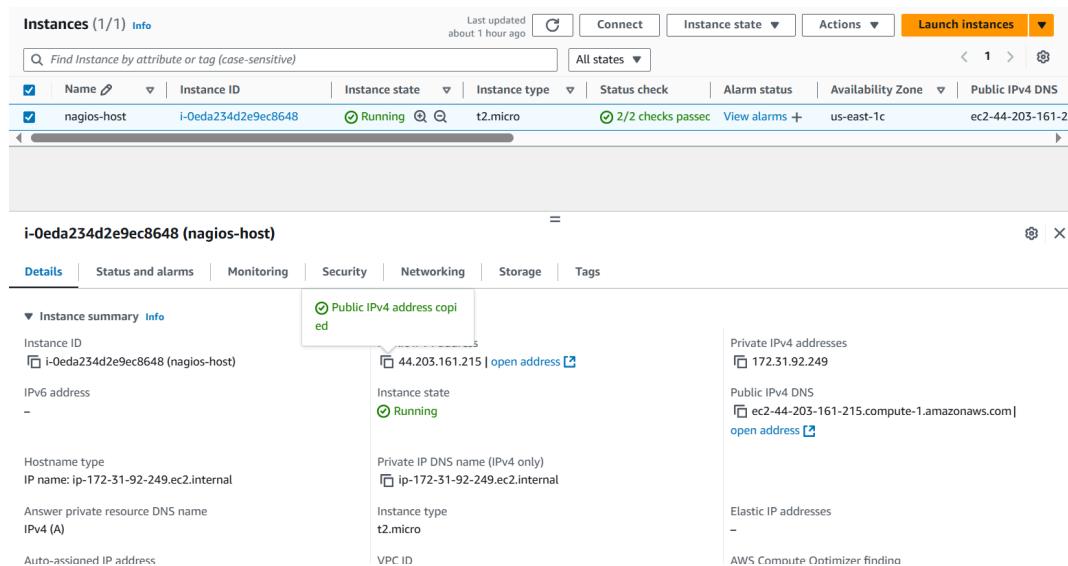
```
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo nano /etc/init.d/nagios
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo chmod +x /etc/init.d/nagios
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo chkconfig --add nagios
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo chkconfig nagios on
Note: Forwarding request to 'systemctl enable nagios.service'.
Synchronizing state of nagios.service with sysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable nagios
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service → /usr/lib/systemd/system/nagios.service.
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$
```

`sudo service nagios start`

```
[ec2-user@ip-172-31-92-249 nagios-plugins-2.4.11]$ sudo service nagios start
Starting Nagios...
Nagios Core 4.5.5
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2024-09-17
License: GPL

Website: https://www.nagios.org
Nagios 4.5.5 starting... (PID=72261)
Local time is Tue Oct 01 20:59:58 UTC 2024.
wproc: Successfully registered manager as @wproc with query handler
wproc: Registry request: name=Core Worker 72265;pid=72265
wproc: Registry request: name=Core Worker 72264;pid=72264
wproc: Registry request: name=Core Worker 72263;pid=72263
wproc: Registry request: name=Core Worker 72262;pid=72262
Successfully launched command file worker with pid 72266
wproc: NOTIFY job 4 from worker Core Worker 72262 is a non-check helper but exited with return code 127
wproc: host=localhost; service=Swap Usage; contact=nagiosadmin
wproc: early_timeout=0; exited_ok=1; wait_status=32512; error_code=0;
wproc: stderr line 01: /bin/sh: line 1: /bin/mail: No such file or directory
wproc: stderr line 02: /usr/bin/printf: write error: Broken pipe
```

Get your public IPv4 address from your instance. We will require it for connecting to our nginx server



Browse for this url: http://<your_public_ip_address>/nagios

The browser may ask you for your nagios credentials which set in the earlier steps

The username is nagiosadmin and enter the password that you set earlier

The screenshot shows the Nagios Core web interface. The top navigation bar indicates the URL is 34.229.45.75/nagios and the connection is not secure. The main header features the Nagios Core logo with the text "Nagios® Core™ Version 4.5.5" and the date "September 17, 2024". A green checkmark icon with the text "Process running with PID 62668" is displayed. The left sidebar contains a navigation menu with sections like General, Current Status, Service Groups, Problems, Reports, and Configuration. The "Current Status" section is expanded, showing links for Tactical Overview, Map, Hosts, Services, Host Groups, Summary, Grid, and Service Groups. The "Service Groups" section is also expanded, showing links for Summary and Grid. The "Problems" section shows 0 services and 0 hosts. The "Reports" section shows links for Availability, Trends, Alerts, History, Summary, Histogram, Notifications, and Event Log. The main content area includes a "Get Started" box with a bulleted list of steps, a "Latest News" box, and a "Don't Miss..." box. A "Quick Links" box on the right provides links to Nagios Library, Nagios Labs, Nagios Exchange, Nagios Support, Nagios.com, and Nagios.org. A vertical "Page Tour" button is located on the far right.

Conclusion:

In this experiment, we successfully installed and configured Nagios Core on an Amazon Linux EC2 instance, showcasing its role in continuous monitoring within a DevOps environment. We learned about user management and service configuration, emphasizing Nagios's ability to monitor systems and networks effectively. This experience laid the groundwork for enhancing infrastructure reliability and integrating advanced monitoring strategies in future projects.

Adv DevOps Exp 10

Aim: To perform Port, Service monitoring, Windows/Linux server monitoring using Nagios.

Monitoring Using Nagios:

Step 1: To Confirm Nagios is running on the server side Perform the following command on your Amazon Linux Machine (Nagios-host).

Run this command **sudo systemctl status**

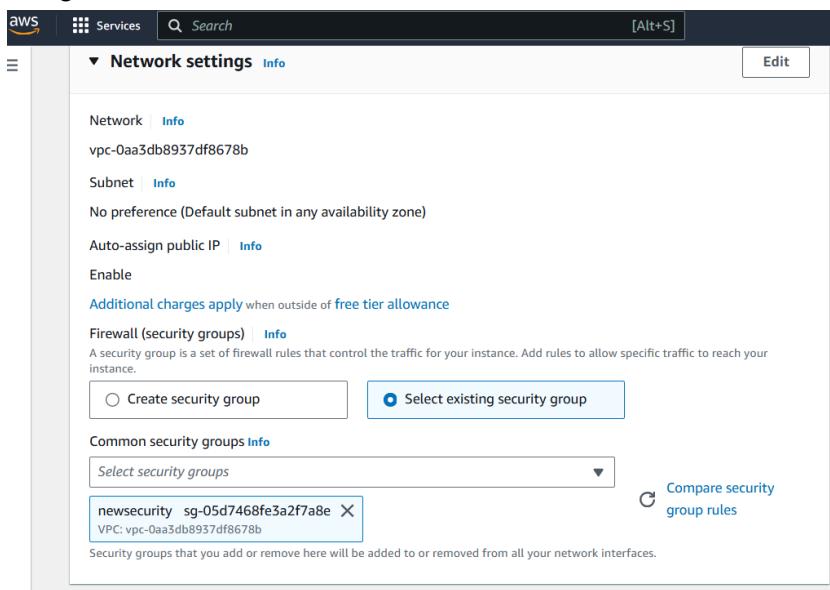
```
ec2-user@ip-172-31-41-160:~$ /downloads/nagios-plugins-2.4.11
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ sudo systemctl status
● ip-172-31-41-160.ec2.internal
    State: running
      Units: 296 loaded (incl. loaded aliases)
        Jobs: 0 queued
       Failed: 0 units
     Since: Wed 2024-10-02 12:28:05 UTC; 33min ago
    Systemd: 252.23-2.amzn2023
   CGroup: /
           └─init.scope
             ├─1 /usr/lib/systemd/systemd --switched-root --system --deserialize=32
             ├─system.slice
             ├─acpid.service
             ├─amazon-ssm-agent.service
             ├─atd.service
             ├─auditd.service
             ├─chronynd.service
             ├─dbus-broker.service
             ├─gssproxy.service
             ├─httpd.service
             ├─libstoragemgmt.service
             └─1940 /usr/bin/lsm -d

1938 /usr/bin/systemd-inhibit --what=handle-suspend-key:handle-hibernate-key --who=noah "--why=acpid instead" --mode=block /usr/sbin/acpid -f
2059 /usr/sbin/acpid -f
2141 /usr/bin/amazon-ssm-agent
2152 /usr/sbin/atd -f
1768 /sbin/auditd
2175 /usr/sbin/chronynd -F 2
1946 /usr/bin/dbus-broker-launch --scope system --audit
1954 dbus-broker --log 4 --controller 9 --machine-id ec2e4d759a3e2f6fe850b14e4cdacabe --max-bytes 536870912 --max-fds 4096 --max-matches 16384 --audit
1959 /usr/sbin/gssproxy -D
49553 /usr/sbin/httpd -DFOREGROUND
49555 /usr/sbin/httpd -DFOREGROUND
49556 /usr/sbin/httpd -DFOREGROUND
49557 /usr/sbin/httpd -DFOREGROUND
49558 /usr/sbin/httpd -DFOREGROUND
62800 /usr/sbin/httpd -DFOREGROUND
```

Step 2: Before we begin,

To monitor a Linux machine, create an **Ubuntu 20.04 server** EC2 Instance in AWS.

Provide it with the **same security group** as the Nagios Host and name it 'nagios-client' alongside the host.



Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

mohit ▼ ⟳ Create new key pair

The screenshot shows the AWS EC2 Instances page. It lists two instances: 'nagios-host' and 'nagios-client', both of which are running. The 'Instances' section is expanded in the sidebar. The main table includes columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 DNS. The 'nagios-host' instance has an instance ID of i-03facef442a77494d and is located in us-east-1a. The 'nagios-client' instance has an instance ID of i-0b934b61f21351c1b and is also located in us-east-1a.

Step 3: TO BE DONE IN THE Nagios-host TERMINAL

In the nagios-host terminal, run this command

ps -ef | grep nagios

```
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ ps -ef | grep nagios
ec2-user 63115 2315 0 13:03 pts/0 00:00:00 grep --color=auto nagios
[ec2-user@ip-172-31-41-160 nagios-plugins-2.4.11]$ ■
```

To become a root user, run '**sudo su**' and make two directories using the following commands. If one is running these commands in windows powershell, make sure that he/she copies it line by line as powershell might make an error while interpreting multiple lines

mkdir /usr/local/nagios/etc/objects/monitorhosts

mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts

```
[ec2-user@ip-172-31-92-249 ~]$ sudo su
[root@ip-172-31-92-249 ec2-user]# mkdir /usr/local/nagios/etc/objects/monitorhosts
[root@ip-172-31-92-249 ec2-user]# mkdir /usr/local/nagios/etc/objects/monitorhosts/linuxhosts
[root@ip-172-31-92-249 ec2-user]#
```

Copy the sample localhost.cfg file to linuxhost folder. Use the following mentioned command to achieve it

cp /usr/local/nagios/etc/objects/localhost.cfg

/usr/local/nagios/etc/objects/monitorhosts/linuxserver.cfg

Open linuxserver.cfg using nano and make the following changes. This is a conf type file in which we will have to modify the configurations in way which will help us specify the hosts and clients to be monitored

nano /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg

Changes to be made:

1. Change the hostname to linux-server (EVERYWHERE ON THE FILE)
2. Change address to the public IP address of your LINUX CLIENT.
3. Change hostgroup_name under hostgroup to linux-servers1

```
# HOST DEFINITION
#
#####
# Define a host for the local machine

define host {
    use          linux-server           ; Name of host template to use
                                         ; This host definition will inherit all variables that are defined
                                         ; in (or inherited by) the linux-server host template definition.

    host_name    linux-server
    alias        localhost
    address     54.172.92.226
}

#####
# Define an optional hostgroup for Linux machines

define hostgroup {
    hostgroup_name   linux-servers1    ; The name of the hostgroup
    alias            Linux Servers      ; Long name of the group
    members          localhost         ; Comma separated list of hosts that belong to this group
}
```

IMP: Everywhere else on the file, change the hostname to linux-server instead of localhost.

Open the Nagios Config file and add the following line

nano /usr/local/nagios/etc/nagios.cfg

Add the following line in the file and save

cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/

```
# OBJECT CONFIGURATION FILE(S)
# These are the object configuration files in which you define hosts,
# host groups, contacts, contact groups, services, etc.
# You can split your object definitions across several config files
# if you wish (as shown below), or keep them all in a single config file.

# You can specify individual object config files as shown below:
cfg_file=/usr/local/nagios/etc/objects/commands.cfg
cfg_file=/usr/local/nagios/etc/objects/contacts.cfg
cfg_file=/usr/local/nagios/etc/objects/timeperiods.cfg
cfg_file=/usr/local/nagios/etc/objects/templates.cfg

# Definitions for monitoring the local (Linux) host
cfg_file=/usr/local/nagios/etc/objects/localhost.cfg
cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/
# Definitions for monitoring a Windows machine
#cfg_file=/usr/local/nagios/etc/objects/windows.cfg
```

Verify the configuration files by running the following command

/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

```
[root@ip-172-31-41-160 nagios-plugins-2.4.11]# /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

Nagios Core 4.5.5
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2024-09-17
License: GPL

Website: https://www.nagios.org
Reading configuration data...
  Read main config file okay...
  Read object config files okay...

Running pre-flight check on configuration data...

Checking objects...
  Checked 16 services.
  Checked 2 hosts.
  Checked 2 host groups.
  Checked 0 service groups.
  Checked 1 contacts.
  Checked 1 contact groups.
  Checked 24 commands.
  Checked 5 time periods.
  Checked 0 host escalations.
  Checked 0 service escalations.
Checking for circular paths...
  Checked 2 hosts
  Checked 0 service dependencies
  Checked 0 host dependencies
  Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors: 0

Things look okay - No serious problems were detected during the pre-flight check
[root@ip-172-31-41-160 nagios-plugins-2.4.11]#
```

You are good to go if there are no errors.

Restart the nagios service

service nagios restart

And by running sudo systemctl status nagios, we can again check whether our server is running or not

```

root@ip-172-31-41-160:/tmp/nagios-plugins-2.4.11]
[root@ip-172-31-41-160 nagios-plugins-2.4.11]# sudo systemctl restart nagios
[root@ip-172-31-41-160 nagios-plugins-2.4.11]# sudo systemctl status nagios
● nagios.service - Nagios Core 5
   Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Wed 2024-10-02 13:20:17 UTC; 7s ago
       Docs: https://www.nagios.org/documentation
   Process: 78776 ExecStart=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
  Process: 78777 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
 Main PID: 78778 (nagios)
   Tasks: 6 (limit: 1112)
      Memory: 4.0M
        CPU: 24ms
      CGroup: /system.slice/nagios.service
          └─78778 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Oct 02 13:20:17 ip-172-31-41-160.ec2.internal nagios[78778]: qh: echo service query handler registered
Oct 02 13:20:17 ip-172-31-41-160.ec2.internal nagios[78778]: qh: help for the query handler registered
Oct 02 13:20:17 ip-172-31-41-160.ec2.internal nagios[78778]: wpoc: Successfully registered Nagios @proc with query handler
Oct 02 13:20:17 ip-172-31-41-160.ec2.internal nagios[78778]: wpoc: Registry request: name=Core Worker 78782;pid=78782
Oct 02 13:20:17 ip-172-31-41-160.ec2.internal nagios[78778]: wpoc: Registry request: name=Core Worker 78781;pid=78781
Oct 02 13:20:17 ip-172-31-41-160.ec2.internal nagios[78778]: wpoc: Registry request: name=Core Worker 78780;pid=78780
Oct 02 13:20:17 ip-172-31-41-160.ec2.internal nagios[78778]: wpoc: Registry request: name=Core Worker 78779;pid=78779
Oct 02 13:20:17 ip-172-31-41-160.ec2.internal nagios[78778]: Successfully launched command file worker with pid 78783
Oct 02 13:20:21 ip-172-31-41-160.ec2.internal nagios[78778]: HOST ALERT: linux-server;UP;SOFT;1;PING OK - Packet loss = 0%, RTA = 0.93 ms
Oct 02 13:20:24 ip-172-31-41-160.ec2.internal nagios[78778]: SERVICE ALERT: localhost;HTTP;WARNING;HARD;4;HTTP WARNING: HTTP/1.1 403 Forbidden - 319 bytes in 0.0
[root@ip-172-31-41-160 nagios-plugins-2.4.11]# sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
     Drop-In: /usr/lib/systemd/system/httpd.service.d
       └─php-fpm.conf
     Active: active (running) since Wed 2024-10-02 12:47:56 UTC; 33min ago
       Docs: man:httpd.service(8)
   Main PID: 49553 (httpd)
     Status: "Total requests: 26; Idle/Busy workers 100/0;Requests/sec: 0.0129; Bytes served/sec: 94 B/sec"
     Tasks: 230 (limit: 1112)
        Memory: 1.7M
          CPU: 1.416s
        CGroup: /system.slice/httpd.service
            ├─49553 /usr/sbin/httpd -DFOREGROUND

```

Step 4: TO BE DONE IN THE Nagios-client TERMINAL

Now it is time to switch to the client machine.

SSH into the machine or simply use the EC2 Instance Connect feature.

```

PS C:\WINDOWS\system32> cd C:\Users\DEll\Downloads
PS C:\Users\DEll\Downloads> ssh -i "mohit.pem" ubuntu@ec2-54-172-92-226.compute-1.amazonaws.com
The authenticity of host 'ec2-54-172-92-226.compute-1.amazonaws.com (54.172.92.226)' can't be established.
ECDSA key fingerprint is SHA256:e/WkFQRuHSpjQ5hDMA0dku8msNHETN9SAgzEy53E.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-172-92-226.compute-1.amazonaws.com,54.172.92.226' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Wed Oct  2 13:26:11 UTC 2024

System load:  0.0           Processes:      104
Usage of /:   22.8% of  6.71GB  Users logged in:    0
Memory usage: 20%           IPv4 address for enx0: 172.31.36.100
Swap usage:   0%

* Ubuntu Pro delivers the most comprehensive open source security and
  compliance features.

  https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
law.

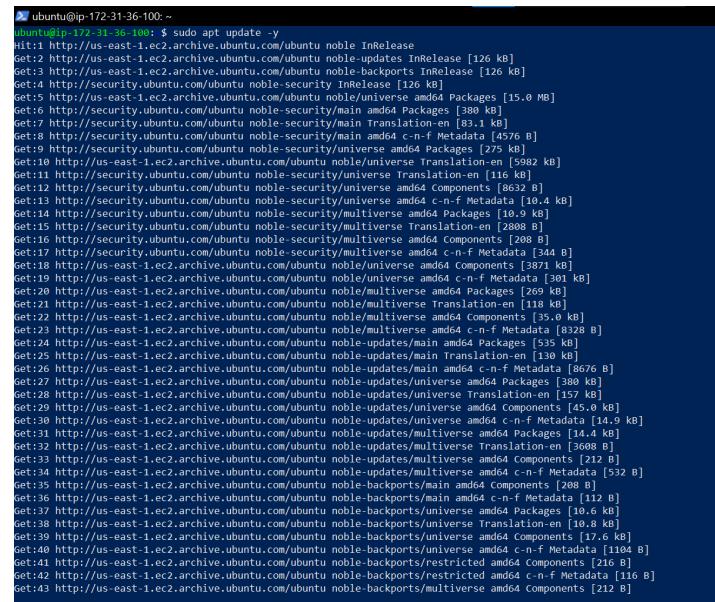
```

Make a package index update and install gcc, nagios-nrpe-server and the plugins. Run the following commands to achieve the same.

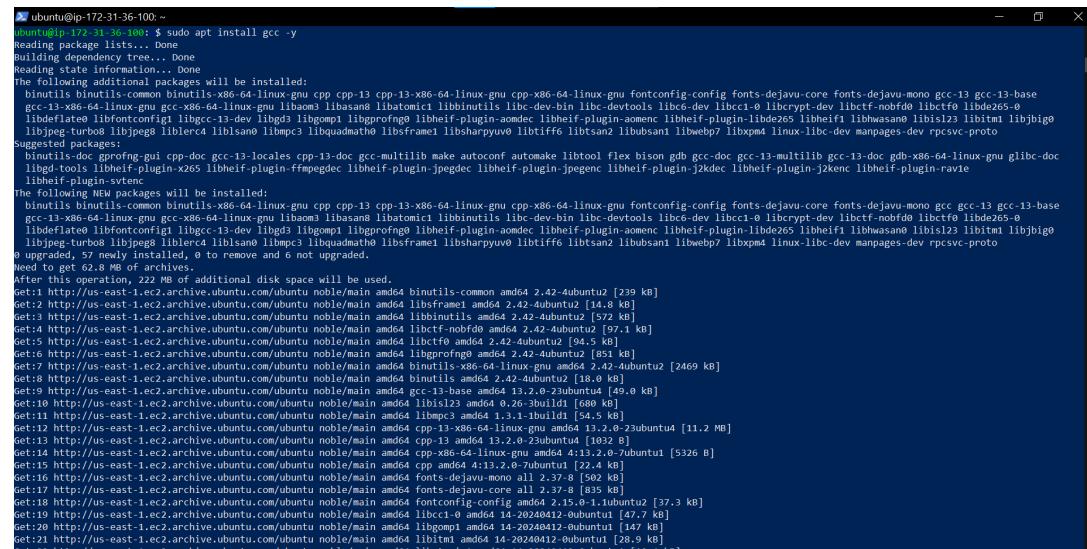
sudo apt update -y

sudo apt install gcc -y

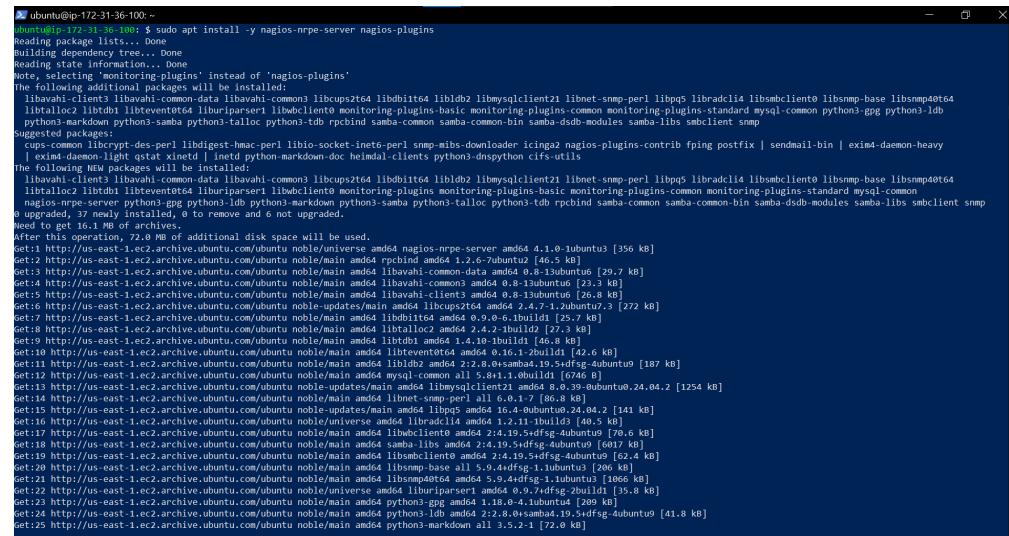
sudo apt install -y nagios-nrpe-server nagios-plugins



A terminal window showing the output of the command \$ sudo apt update -y. The output lists numerous packages from various repositories (main, universe, multiverse, restricted) and architectures (amd64, i386) being updated.



A terminal window showing the installation of the gcc package using \$ sudo apt install gcc -y. The process shows dependency resolution and the download of multiple packages from the main repository.



A terminal window showing the installation of nagios nrpe-server and nagios-plugins using \$ sudo apt install -y nagios-nrpe-server nagios-plugins. The process shows the download and installation of several configuration files and libraries from the main repository.

Open nrpe.cfg file to make changes.

sudo nano /etc/nagios/nrpe.cfg

Under allowed_hosts, add your nagios host IP address like so

```
ubuntu@ip-172-31-36-100: ~
GNU nano 7.2

#
# Note: The daemon only does rudimentary checking of the client's IP
# address. I would highly recommend adding entries in your /etc/hosts.allow
# file to allow only the specified host to connect to the port
# you are running this daemon on.
#
# NOTE: This option is ignored if NRPE is running under either inetd or xinetd

allowed_hosts=127.0.0.1,34.229.45.75

#
# COMMAND ARGUMENT PROCESSING
# This option determines whether or not the NRPE daemon will allow clients
# to specify arguments to commands that are executed. This option only works
# if the daemon was configured with the --enable-command-args configure script
```

Now restart the NRPE server by this command.

sudo systemctl restart nagios-nrpe-server

```
ubuntu@ip-172-31-36-100: ~$ sudo systemctl restart nagios-nrpe-server
ubuntu@ip-172-31-36-100: ~$
```

Run the following command in the Nagios-host terminal

sudo systemctl status nagios

```
[root@ip-172-31-41-160 nagios-plugins-2.4.11]# sudo systemctl status nagios
● nagios.service - Nagios Core 4.5.5
   Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
   Active: active (running) since Wed 2024-10-02 13:20:17 UTC; 15min ago
     Docs: https://www.nagios.org/documentation
 Main PID: 78778 (nagios)
   Tasks: 6 (limit: 1112)
    Memory: 4.3M
       CPU: 403ms
      CGroup: /system.slice/nagios.service
              └─78778 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
                  ├─78779 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
                  ├─78780 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
                  ├─78781 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
                  ├─78782 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
                  └─78783 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Oct 02 13:22:54 ip-172-31-41-160.ec2.internal nagios[78778]: SERVICE NOTIFICATION: nagiosadmin;localhost;Swap Usage;CRITICAL;notify-service-by-email;SWAP CRITICAL - 0% free (0 MB out of 0 MB)
Oct 02 13:22:54 ip-172-31-41-160.ec2.internal nagios[78778]: wproc: NOTIFY job 3 from worker Core Worker 78782 is a non-check helper but exited with return code 127
Oct 02 13:22:54 ip-172-31-41-160.ec2.internal nagios[78778]: wproc: host=localhost; service=Swap Usage; contact=nagiosadmin
Oct 02 13:22:54 ip-172-31-41-160.ec2.internal nagios[78778]: wproc: early timeout=0; exited_ok=1; wait_status=2512; error_code=0;
Oct 02 13:22:54 ip-172-31-41-160.ec2.internal nagios[78778]: wproc: stderr line 01: /bin/sh: line 1: /bin/mail: No such file or directory
Oct 02 13:22:54 ip-172-31-41-160.ec2.internal nagios[78778]: wproc: stderr line 02: /usr/bin/printf: write error: Broken pipe
Oct 02 13:23:13 ip-172-31-41-160.ec2.internal nagios[78778]: SERVICE ALERT: linux-server;Total Processes;OK;HARD;1;PROCS OK: 37 processes with STATE = RZDT
Oct 02 13:23:50 ip-172-31-41-160.ec2.internal nagios[78778]: SERVICE ALERT: linux-server;Current Load;OK;HARD;1;OK - load average: 0.01, 0.07, 0.04
Oct 02 13:24:28 ip-172-31-41-160.ec2.internal nagios[78778]: SERVICE ALERT: linux-server;Current Users;OK;HARD;1;USERS OK - 2 users currently logged in
Oct 02 13:24:46 ip-172-31-41-160.ec2.internal nagios[78778]: SERVICE ALERT: localhost;Current Users;OK;HARD;1;USERS OK - 2 users currently logged in
Lines 1-26/26 (END)
```

Step 5: Visiting your nagios server using your nagios-host ip address

Open up your browser and look for http://<public_ip_address_of_nagios-host>/nagios

The screenshot shows the Nagios Core 4.5.5 dashboard. At the top right, it displays "Nagios® Core™ Version 4.5.5" and the date "September 17, 2024". A green checkmark indicates "Daemon running with PID 78778". The left sidebar contains navigation links for General, Current Status, and Reports. The main content area includes sections for "Get Started", "Latest News", and "Don't Miss...".

Click on Hosts.

The screenshot shows the "Host Status Details For All Host Groups" section. It displays two hosts: "linux-server" and "localhost", both marked as "UP". The "Status Information" column shows PING OK for both hosts. On the far right, there is a "Page Tour" button.

Click on linux-server to view host information

The screenshot shows the Nagios web interface at <http://34.229.45.75/nagios/>. The left sidebar is collapsed, showing the main navigation menu. The central panel displays 'Host Information' for 'localhost'. Key details include:

- Last Updated: Wed Oct 2 13:40:56 UTC 2024
- Updated every 90 seconds
- Nagios® Core™ 4.5.5 - www.nagios.org
- Logged in as `nagiosadmin`
- Host: `localhost` (`linux-server`)
- Member of: **No hostgroups**
- IP: 54.172.92.226

Host State Information table:

Host Status:	UP (for 0d 0h 20m 39s)
Status Information:	PING OK - Packet loss = 0%, RTA = 0.84 ms
Performance Data:	rta=0.838000ms;3000.000000;5000.000000;0.000000 pl=0%;80;100;0
Current Attempt:	1/10 (HARD state)
Last Check Time:	10-02-2024 13:40:17
Check Type:	ACTIVE
Check Latency / Duration:	0.000 / 4.121 seconds
Next Scheduled Active Check:	10-02-2024 13:45:17
Last State Change:	10-02-2024 13:20:17
Last Notification:	N/A (notification 0)
In This Host Flapping?	NO (0.00% state change)
In Scheduled Downtime?	NO
Last Update:	10-02-2024 13:40:46 (0d 0h 0m 10s ago)

Active Checks: Enabled

Host Commands sidebar (partial list):

- Locate host on map
- Disable active checks of this host
- Re-schedule the next check of this host
- Submit passive check result for this host
- Stop accepting passive checks for this host
- Stop obsessing over this host
- Disable notifications for this host
- Send custom host notification
- Schedule downtime for this host
- Schedule downtime for all services on this host
- Disable notifications for all services on this host
- Enable notifications for all services on this host
- Schedule a check of all services on this host
- Disable checks of all services on this host
- Enable checks of all services on this host
- Disable event handler for this host
- Disable flap detection for this host
- Clear flapping state for this host

We can even navigate to the services section, which explicitly mentions the status, duration, checks, information about the numerous services present on our hosts

The screenshot shows the Nagios web interface at <http://34.229.45.75/nagios/>. The left sidebar is collapsed. The central panel displays 'Service Status Details For All Hosts' for two hosts: 'linux-server' and 'localhost'. Key details include:

Current Network Status table:

Up	Down	Unreachable	Pending
2	0	0	0

Host Status Totals table:

Ok	Warning	Unknown	Critical	Pending
12	1	0	3	0

Service Status Totals table:

All Problems	All Types
0	2

Service Status Details For All Hosts table:

Host	Service	Status	Last Check	Duration	Attempt	Status Information
linux-server	Current Load	OK	10-02-2024 13:38:50	0d 0h 18m 19s	1/4	OK - load average: 0.00, 0.00, 0.00
	Current Users	OK	10-02-2024 13:39:28	0d 0h 17m 41s	1/4	USERS OK - 3 users currently logged in
	HTTP	CRITICAL	10-02-2024 13:40:05	0d 0h 27m 4s	4/4	connect to address 54.172.92.226 and port 80: Connection refused
	PING	OK	10-02-2024 13:40:43	0d 0h 21m 26s	1/4	PING OK - Packet loss = 0%, RTA = 1.05 ms
	Root Partition	OK	10-02-2024 13:41:20	0d 0h 20m 49s	1/4	DISK OK - free space: / 6122 MB (75.43% inode=98%)
	SSH	OK	10-02-2024 13:41:58	0d 0h 20m 11s	1/4	SSH OK - OpenSSH_9.6p1 Ubuntu-3ubuntu13.5 (protocol 2.0)
	Swap Usage	CRITICAL	10-02-2024 13:37:35	0d 0h 24m 34s	4/4	SWAP CRITICAL - 0% free (0 MB out of 0 MB) - Swap is either disabled, not present, or of zero size.
	Total Processes	OK	10-02-2024 13:38:13	0d 0h 18m 56s	1/4	PROCS OK - 38 processes with STATE = RSZDT
	Current Load	OK	10-02-2024 13:40:09	0d 0h 22m 0s	1/4	OK - load average: 0.00, 0.00, 0.00
	Current Users	OK	10-02-2024 13:39:46	0d 0h 17m 23s	1/4	USERS OK - 3 users currently logged in
localhost	HTTP	WARNING	10-02-2024 13:40:24	0d 0h 21m 45s	4/4	HTTP WARNING: HTTP/1.1 403 Forbidden - 319 bytes in 0.001 second response time
	PING	OK	10-02-2024 13:41:01	0d 0h 21m 8s	1/4	PING OK - Packet loss = 0%, RTA = 0.04 ms
	Root Partition	OK	10-02-2024 13:41:39	0d 0h 20m 30s	1/4	DISK OK - free space: / 6122 MB (75.43% inode=98%)
	SSH	OK	10-02-2024 13:37:16	0d 0h 19m 53s	1/4	SSH OK - OpenSSH_8.7 (protocol 2.0)
	Swap Usage	CRITICAL	10-02-2024 13:37:54	0d 0h 24m 15s	4/4	SWAP CRITICAL - 0% free (0 MB out of 0 MB) - Swap is either disabled, not present, or of zero size.
	Total Processes	OK	10-02-2024 13:42:01	0d 0h 20m 8s	1/4	PROCS OK - 38 processes with STATE = RSZDT

Results 1 - 16 of 16 Matching Services

Conclusion: In conclusion, the experiment focused on monitoring ports, services, and a Linux server using Nagios. Through the step-by-step process, we successfully configured Nagios to monitor essential network services on the Linux server. By setting up both the Nagios host and client, we were able to track system performance, ensure service availability, and monitor key metrics like CPU and memory usage.

Adv DevOps Exp 11

Aim: To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

Theory:

AWS Lambda

AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS). Users of AWS Lambda create functions, self-contained applications written in one of the supported languages and runtimes, and upload them to AWS Lambda, which executes those functions in an efficient and flexible manner. The Lambda functions can perform any kind of computing task, from serving web pages and processing streams of data to calling APIs and integrating with other AWS services. The concept of “serverless” computing refers to not needing to maintain your own servers to run these functions. AWS Lambda is a fully managed service that takes care of all the infrastructure for you. And so “serverless” doesn’t mean that there are no servers involved: it just means that the servers, the operating systems, the network layer and the rest of the infrastructure have already been taken care of so that you can focus on writing application code.

Features of AWS Lambda

- AWS Lambda easily scales the infrastructure without any additional configuration. It reduces the operational work involved.
- It offers multiple options like AWS S3, CloudWatch, DynamoDB, API Gateway, Kinesis, CodeCommit, and many more to trigger an event.
- You don't need to invest upfront. You pay only for the memory used by the lambda function and minimal cost on the number of requests hence cost-efficient.
- AWS Lambda is secure. It uses AWS IAM to define all the roles and security policies.
- It offers fault tolerance for both services running the code and the function. You do not have to worry about the application down.

Packaging Functions

Lambda functions need to be packaged and sent to AWS. This is usually a process of compressing the function and all its dependencies and uploading it to an S3 bucket. And letting AWS know that you want to use this package when a specific event takes place. To help us with this process we use the Serverless Stack Framework (SST). We'll go over this in detail later on in this guide.

Execution Model

The container (and the resources used by it) that runs our function is managed completely by AWS. It is brought up when an event takes place and is turned off if it is not being used. If additional requests are made while the original event is being served, a new container is brought

up to serve a request. This means that if we are undergoing a usage spike, the cloud provider simply creates multiple instances of the container with our function to serve those requests. This has some interesting implications. Firstly, our functions are effectively stateless. Secondly,

each request (or event) is served by a single instance of a Lambda function. This means that you are not going to be handling concurrent requests in your code. AWS brings up a container whenever there is a new request. It does make some optimizations here. It will hang on to the container for a few minutes (5 - 15mins depending on the load) so it can respond to subsequent requests without a cold start.

Stateless Functions

The above execution model makes Lambda functions effectively stateless. This means that every

time your Lambda function is triggered by an event it is invoked in a completely new environment. You don't have access to the execution context of the previous event.

However, due to the optimization noted above, the actual Lambda function is invoked only once per container instantiation. Recall that our functions are run inside containers. So when a function is first invoked, all the code in our handler function gets executed and the handler function gets invoked. If the container is still available for subsequent requests, your function will get invoked and not the code around it.

For example, the `createNewDbConnection` method below is called once per container instantiation and not every time the Lambda function is invoked. The `myHandler` function on the other hand is called on every invocation.

Common Use Cases for Lambda

Due to Lambda's architecture, it can deliver great benefits over traditional cloud computing setups for applications where:

1. Individual tasks run for a short time;
2. Each task is generally self-contained;
3. There is a large difference between the lowest and highest levels in the workload of the application.

Some of the most common use cases for AWS Lambda that fit these criteria are: Scalable APIs. When building APIs using AWS Lambda, one execution of a Lambda function can serve a single HTTP request. Different parts of the API can be routed to different Lambda functions via Amazon API Gateway. AWS Lambda automatically scales individual functions according to

the demand for them, so different parts of your API can scale differently according to current usage levels. This allows for cost-effective and flexible API setups.

Data processing. Lambda functions are optimized for event-based data processing. It is easy to integrate AWS Lambda with data sources like Amazon DynamoDB and trigger a Lambda function for specific kinds of data events. For example, you could employ Lambda to do some work every time an item in DynamoDB is created or updated, thus making it a good fit for things like notifications, counters and analytics.

Steps to create and test your lambda function are as follows:-

Step 1: Creating a Function inside lambda

Navigate to Lambda inside AWS services and click on 'Create Function' button to create a new function inside Lambda services

The screenshot shows the AWS Lambda service interface. On the left, there's a sidebar with 'Lambda' selected. The main area is titled 'Functions (5)' and lists three functions:

- MainMonitoringFunction**: Description: -, Package type: Zip, Runtime: Python 3.8, Last modified: 3 months ago.
- RedshiftEventSubscription**: Description: Create Redshift event subscription to SNS Topic., Package type: Zip, Runtime: Python 3.8, Last modified: 3 months ago.
- ModLabRole**: Description: updates LabRole to allow it to assume itself, Package type: Zip, Runtime: Python 3.8, Last modified: 3 months ago.

At the top right, there's a 'Create function' button. The bottom of the screen includes standard AWS navigation links like CloudShell, Feedback, and a footer with copyright information.

Inside the Create function process, we are supposed to enter the details such as the name of the function, details regarding the things that we want the function to run or test (eg. programming language).

The screenshot shows the 'Create function' wizard. It starts with a choice between three options:

- Author from scratch**: Start with a simple Hello World example. This option is selected.
- Use a blueprint**: Build a Lambda application from sample code and configuration presets for common use cases.
- Container image**: Select a container image to deploy for your function.

Below this, there's a section for **Basic information** with the following fields:

- Function name**: A text input field containing 'mohit'.
- Runtime**: A dropdown menu set to 'Node.js 20.x'.

The bottom of the screen includes standard AWS navigation links like CloudShell, Feedback, and a footer with copyright information.

Along with these things, we are also required to set permissions for us to access and use the lambda function efficiently. For that purpose, we are to select 'Use an existing role' option and

select 'LabRole' which is an already existing role with permissions to use the Lambda Service.

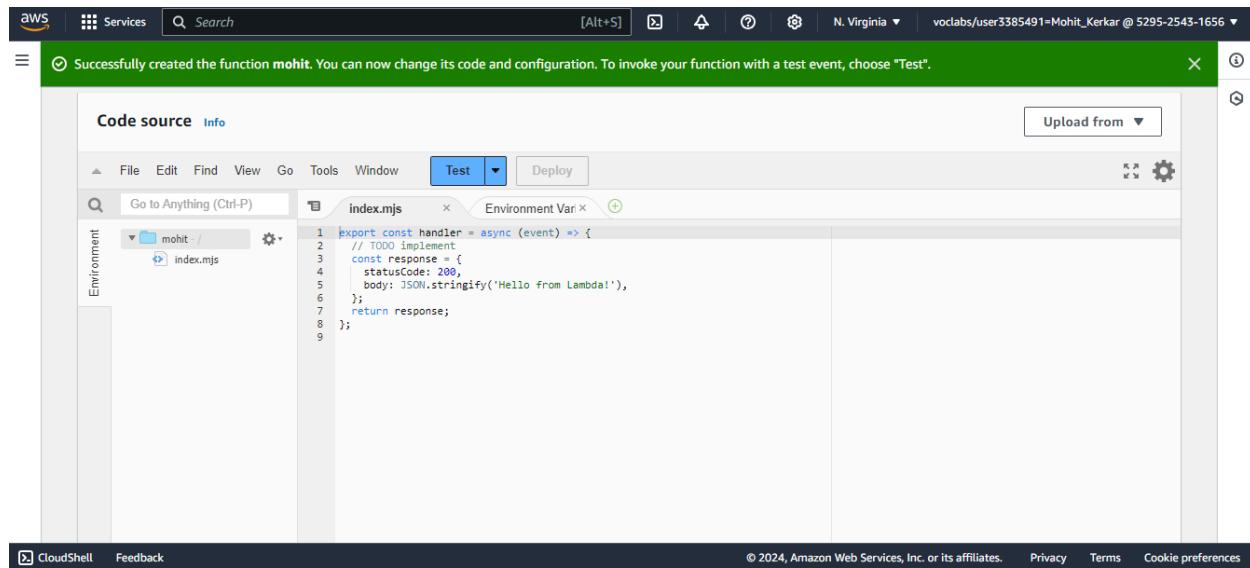
The screenshot shows the AWS Lambda function creation interface. In the 'Execution role' section, the 'Use an existing role' option is selected, and 'LabRole' is chosen from the dropdown menu. Other options like 'Create a new role with basic Lambda permissions' and 'Create a new role from AWS policy templates' are also visible.

And that is how our function inside Lambda is successfully created. This is what appears on the screen after creation of our function (Our dashboard)

The screenshot shows the AWS Lambda function overview page for the 'mohit' function. The top bar indicates success: 'Successfully created the function mohit. You can now change its code and configuration. To invoke your function with a test event, choose "Test".' The main interface includes tabs for 'Function overview' and 'Actions'. The 'Function overview' tab displays the function name 'mohit', a diagram icon, and a 'Layers' section showing '(0)'. Buttons for '+ Add trigger' and '+ Add destination' are present. On the right, there are sections for 'Description', 'Last modified' (12 seconds ago), 'Function ARN' (arn:aws:lambda:us-east-1:529525431656:function:mohit), and 'Function URL'.

Step 2: Modifying function's configuration

Scroll down to view the default code given in the index.mjs file as we see in the screenshot attached below

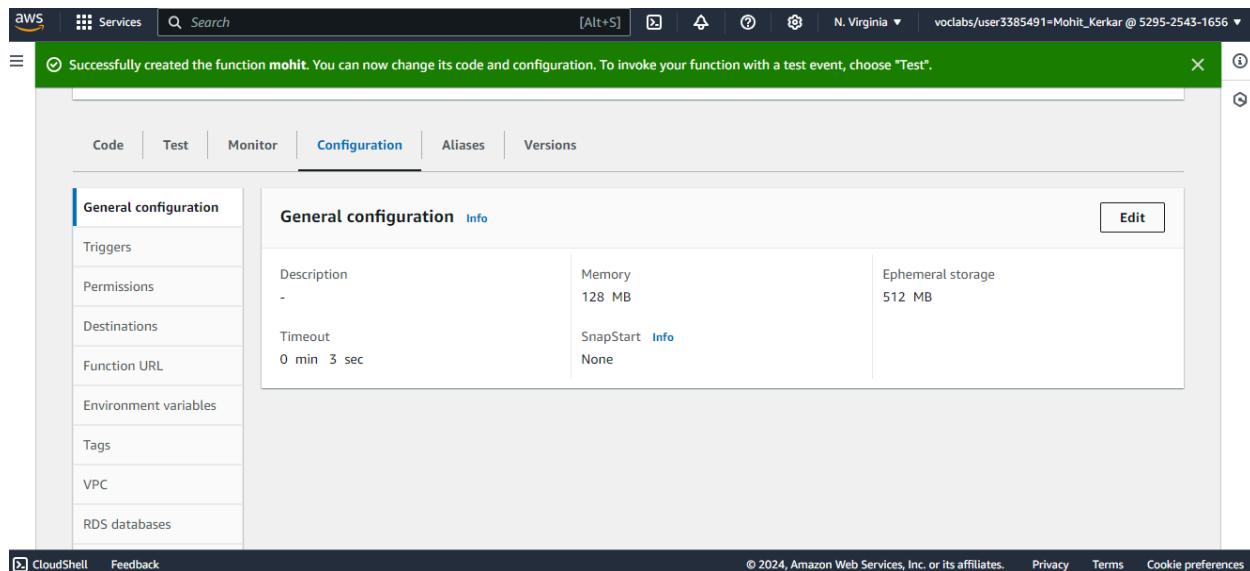


The screenshot shows the AWS Lambda function editor interface. At the top, there's a navigation bar with tabs for Services, Search, and various AWS icons. A green banner at the top right indicates: "Successfully created the function mohit. You can now change its code and configuration. To invoke your function with a test event, choose 'Test'." Below the banner, the main area has tabs for Code source and Info. The Code source tab is active, showing a code editor with the file index.mjs selected. The code in the editor is:

```
1 export const handler = async (event) => {
2     // TODO implement
3     const response = {
4         statusCode: 200,
5         body: JSON.stringify('Hello from Lambda!'),
6     };
7     return response;
8 }
```

On the left, there's a sidebar labeled "Environment" which lists a single folder named "mohit" containing "index.mjs". At the bottom of the editor, there are buttons for Upload from, Test, Deploy, and a gear icon for settings. The footer of the browser window includes links for CloudShell, Feedback, and various AWS terms like Privacy, Terms, and Cookie preferences.

Navigate to the configuration section of our function. This section provides us details about the settings that have been applied to our function for carrying out functionalities like testing on our code in the language that we specified earlier



The screenshot shows the AWS Lambda function configuration page. At the top, there's a navigation bar with tabs for Services, Search, and various AWS icons. A green banner at the top right indicates: "Successfully created the function mohit. You can now change its code and configuration. To invoke your function with a test event, choose 'Test'." Below the banner, the main area has tabs for Code, Test, Monitor, Configuration, Aliases, and Versions. The Configuration tab is active. On the left, a sidebar titled "General configuration" lists options: Triggers, Permissions, Destinations, Function URL, Environment variables, Tags, VPC, and RDS databases. The main panel displays "General configuration" settings with an "Edit" button. The settings shown are:

Description	Memory	Ephemeral storage
-	128 MB	512 MB
Timeout	SnapStart	
0 min 3 sec	Info None	

At the bottom, there are links for CloudShell, Feedback, and various AWS terms like Privacy, Terms, and Cookie preferences.

Click on Edit. Change the timeout to 1 sec to modify the time for the function to be kept running before forcibly terminating it.

After changing the configuration, click on save.

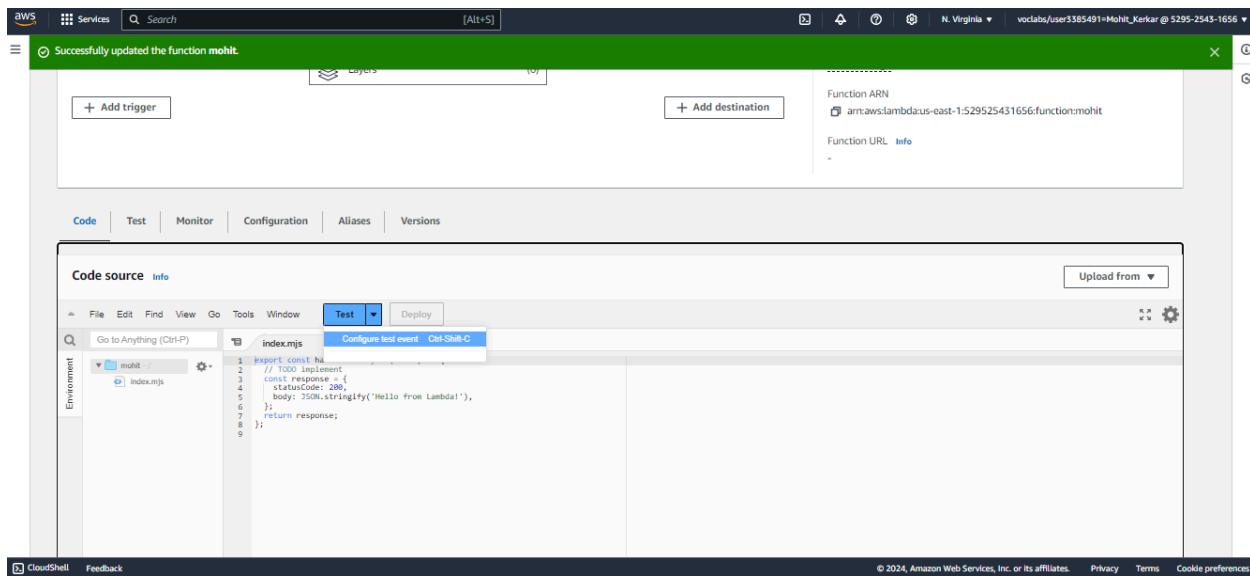
The screenshot shows the 'Edit basic settings' page for the Lambda function 'mohit'. The 'Timeout' field is currently set to 0 min 1 sec. Other settings like Memory (128 MB), Ephemeral storage (512 MB), and SnapStart (None) are also visible.

After saving, the configuration setting changes are reflected in the 'General configuration' section.

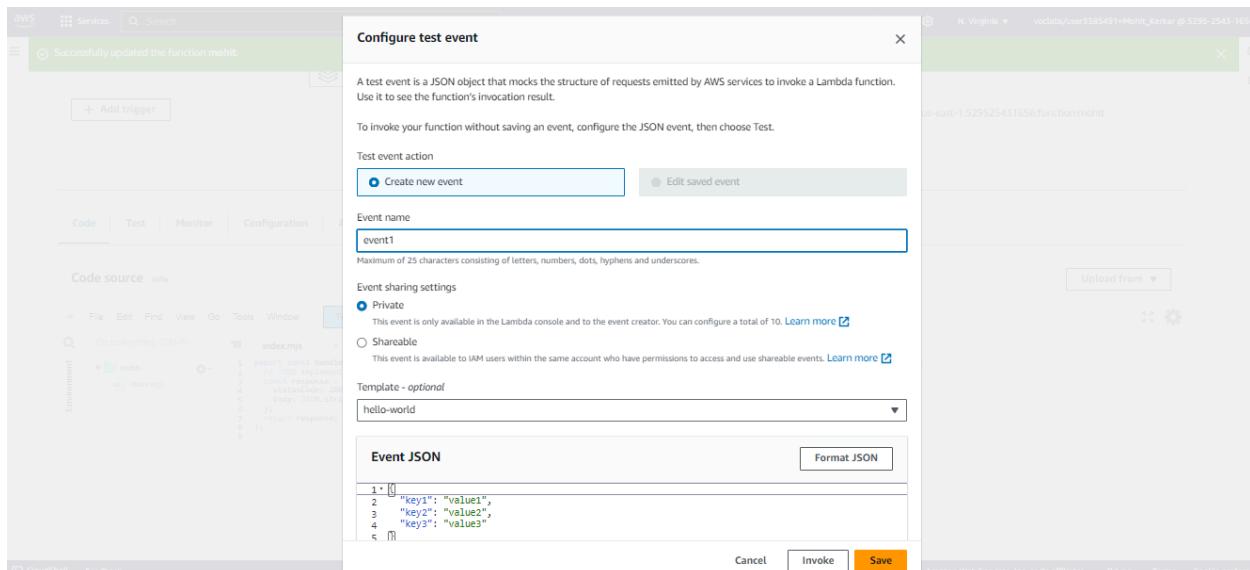
The screenshot shows the 'Configuration' tab for the Lambda function 'mohit'. The 'General configuration' section now shows a Timeout of 0 min 1 sec, reflecting the change made earlier.

Step 3: Creating a new Test event

In our code section, we are given an option to test our code. For this purpose, we are offered numerous test events with which we can possibly test our code.



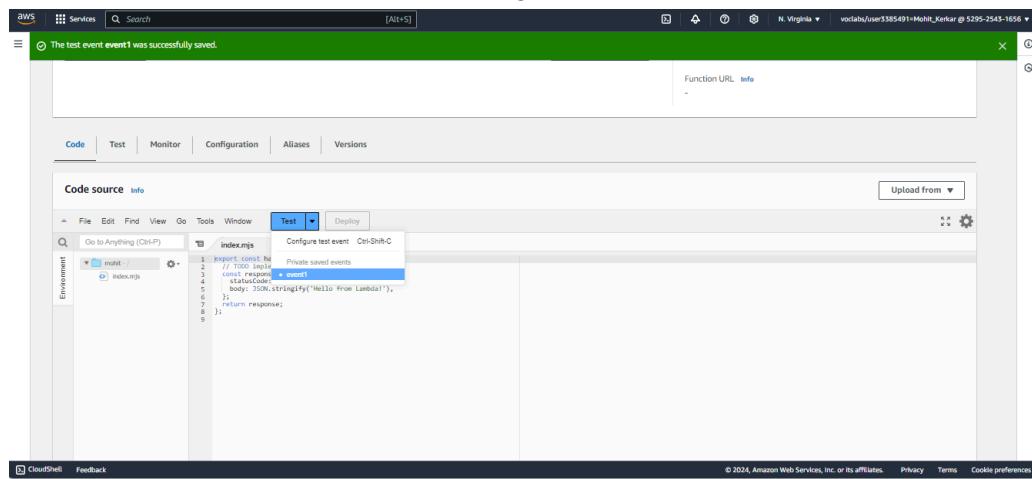
Click on 'configure test event' option for making a new event. Give a suitable name to the event and let other settings be as they are...



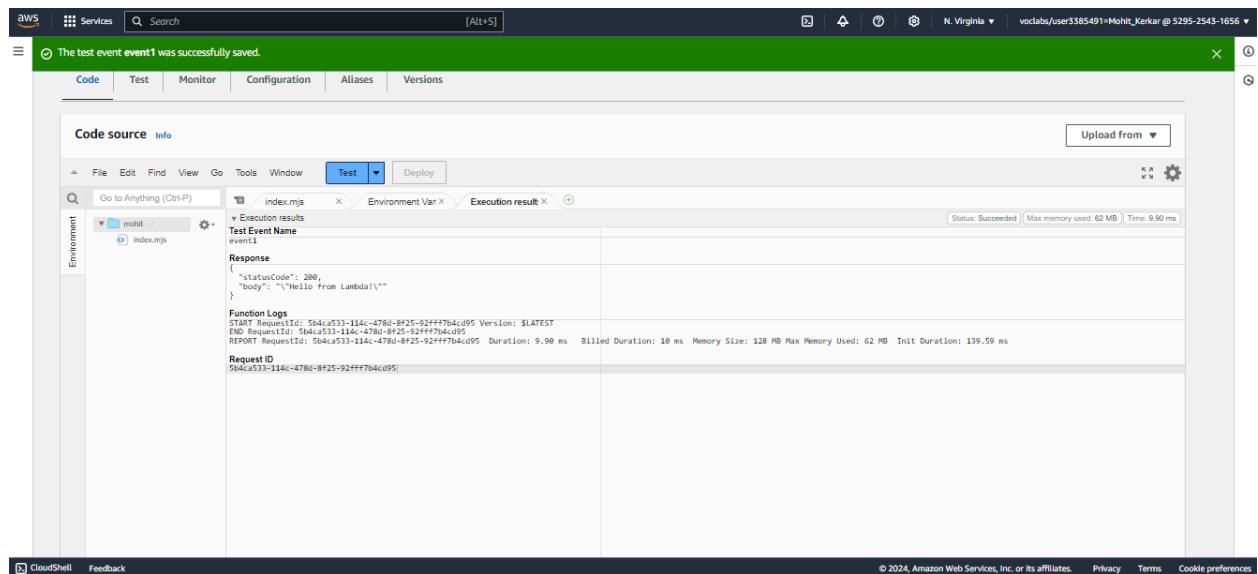
Click on Save..

Step 4: Test with the new test event that we created

Now, in order to test the code according to the test event that we built in the previous step, again select the Test option and within we will be able to see that test event's name (for me it is event1). Select it and observe the changes that happen

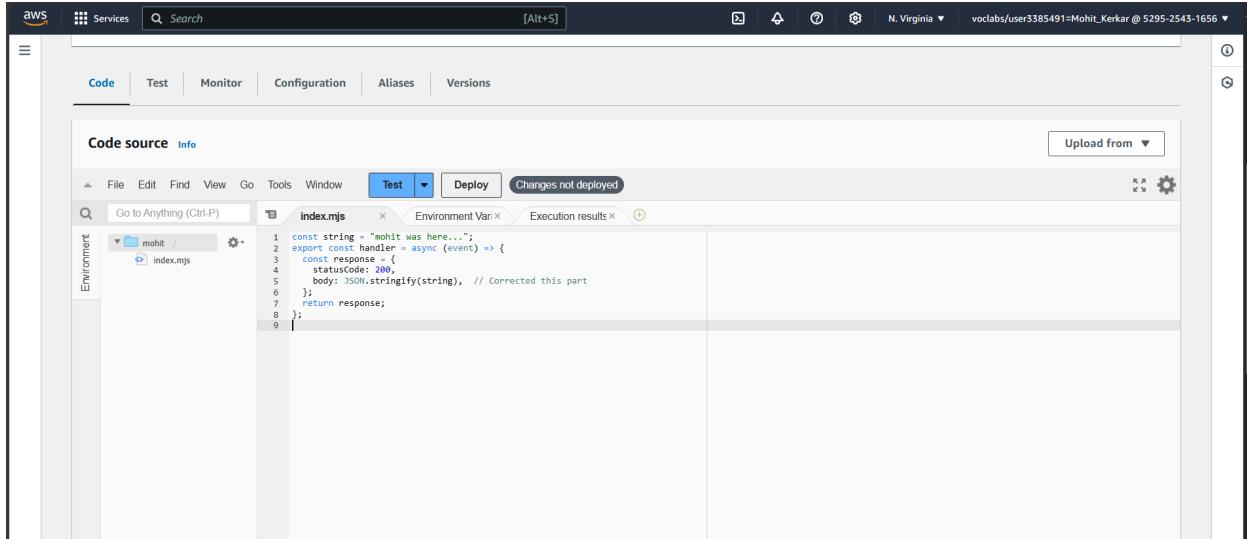


There will be another tab opening which gives us our result after execution. Following is the format with which our Test result comes...



Step 5: Testing our modified code

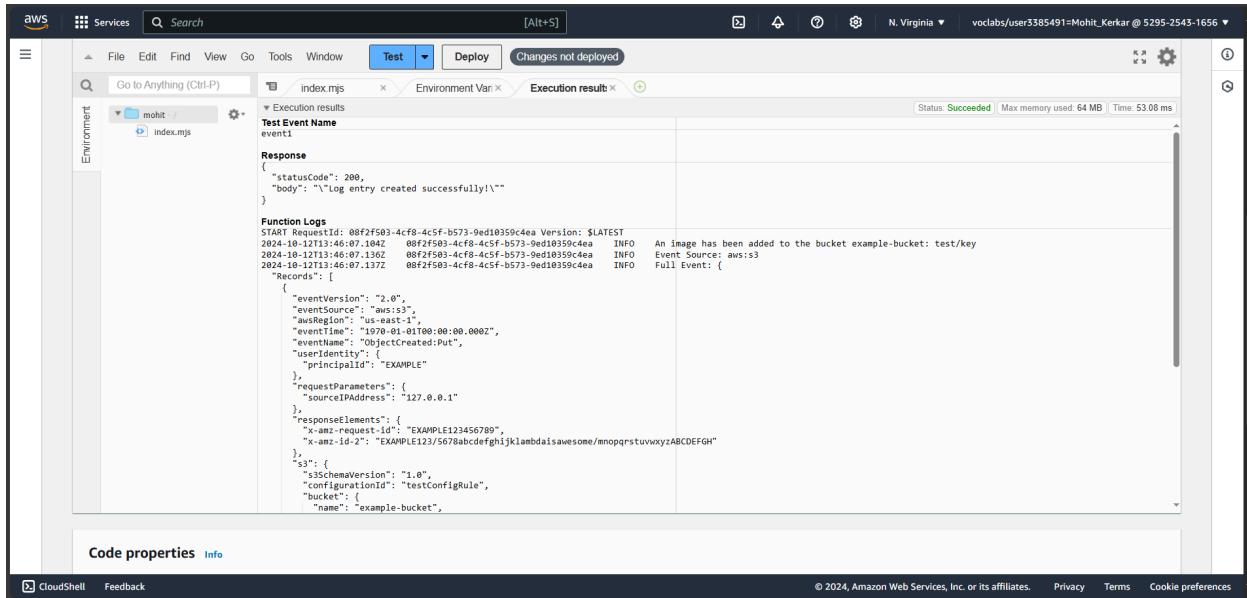
Now, I added a new line at the start of my index.mjs file in my lambda function
 Const string="mohit was here..."



```

1 const string = "mohit was here..."; // Corrected this part
2 export const handler = async (event) => {
3   const response = {
4     statusCode: 200,
5     body: JSON.stringify(string),
6   };
7   return response;
8 }
  
```

The execution result displays that string result in the output as well ..



```

{
  "statusCode": 200,
  "body": "Log entry created successfully!"
}

Function Logs
START RequestId: 00f2f503-4cf8-4c5f-b573-9ed10359c4ea Version: $LATEST
2024-10-12T13:46:07.164Z 00f2f503-4cf8-4c5f-b573-9ed10359c4ea INFO An image has been added to the bucket example-bucket: test/key
2024-10-12T13:46:07.136Z 00f2f503-4cf8-4c5f-b573-9ed10359c4ea INFO Event Source: aws:s3
2024-10-12T13:46:07.137Z 00f2f503-4cf8-4c5f-b573-9ed10359c4ea INFO Full Event: {
  "Records": [
    {
      "eventVersion": "2.0",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "2024-10-12T13:46:07.000Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "responseElements": {
        "x-amz-request-id": "EXAMPLE123456789",
        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklmabaisawesome/mnopqrstuvwxyzABCDEFGH"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "testConfigRule",
        "bucket": {
          "name": "example-bucket"
        }
      }
    }
  ]
}
  
```

Conclusion: In conclusion, AWS Lambda provides an efficient, serverless computing service that enables you to run code without managing infrastructure. By creating and configuring a Lambda function, you can execute code in response to various events, such as S3 uploads or API requests. The setup process involves defining roles, setting up test events, and modifying the function's configuration for specific behaviors like timeouts. AWS Lambda's stateless nature and automatic scaling make it ideal for short-lived, event-driven tasks, such as building APIs or processing data streams, ensuring cost-effective and scalable solutions.

Adv DevOps Exp 12

Aim: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3

Theory:

AWS Lambda and S3 Integration:

AWS Lambda allows you to execute code in response to various events, including those triggered by Amazon S3. When an object is added to an S3 bucket, it can trigger a Lambda function to execute, allowing for event-driven processing without managing servers.

Workflow:

1. Create an S3 Bucket:

- First, create an S3 bucket that will store the objects. This bucket will act as the trigger source for the Lambda function.

2. Create the Lambda Function:

- Set up a new Lambda function using AWS Lambda's console. You can choose a runtime environment like Python, Node.js, or Java.

3. Set Up Permissions:

- Ensure that the Lambda function has the necessary permissions to access S3. You can do this by attaching an IAM role with policies that allow reading from the bucket and writing logs to CloudWatch.

4. Configure S3 Trigger:

- Link the S3 bucket to the Lambda function by setting up a trigger. Specify that the function should be triggered when an object is created in the bucket (e.g., when an image is uploaded).

5. Test the Setup:

- Upload an object (e.g., an image) to the S3 bucket to test the trigger. The Lambda function should execute and log the message “An Image has been added” in AWS CloudWatch Logs.

Step 1: S3 bucket creation

Navigate to the S3 inside services section on AWS and create a new bucket

The screenshot shows the AWS S3 Buckets page. On the left, there is a navigation sidebar with links for Buckets, Access Grants, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, IAM Access Analyzer for S3, Storage Lens, Dashboards, Storage Lens groups, AWS Organizations settings, Feature spotlight, and AWS Marketplace for S3. The main content area shows an account snapshot and a general purpose buckets table. The table has one row with the following data:

Name	AWS Region	IAM Access Analyzer	Creation date
elasticbeanstalk-us-east-1-529525431656	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 5, 2024, 14:26:01 (UTC+05:30)

At the bottom of the page, there are links for CloudShell, Feedback, and a footer with copyright information and links for Privacy, Terms, and Cookie preferences.

Give some suitable name to the bucket

General configuration

AWS Region: US East (N. Virginia) us-east-1

Bucket type: General purpose

Bucket name: lambdabucket12

Object Ownership: Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

Unselect all of the default settings related to 'Block public access'

Block all public access

Block public access to buckets and objects granted through new access control lists (ACLs)

Block public access to buckets and objects granted through any access control lists (ACLs)

Block public access to buckets and objects granted through new public bucket or access point policies

Block public and cross-account access to buckets and objects through any public bucket or access point policies

⚠ Turning off block all public access might result in this bucket and the objects within becoming public

AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

I acknowledge that the current settings might result in this bucket and the objects within becoming public.

And here, you have your new S3 bucket created...

Account snapshot - updated every 24 hours All AWS Regions

General purpose buckets | Directory buckets

Name	AWS Region	IAM Access Analyzer	Creation date
elasticbeanstalk-us-east-1-529525431656	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 5, 2024, 14:26:01 (UTC+05:30)
lambdabucket12	US East (N. Virginia) us-east-1	View analyzer for us-east-1	October 7, 2024, 15:06:22 (UTC+05:30)

Step 2: Upload an image onto our bucket

Now, open the bucket and click on upload

The screenshot shows the AWS S3 console interface. The top navigation bar includes 'Services', 'Search', and 'N. Virginia'. Below it, the breadcrumb path shows 'Amazon S3 > Buckets > lambdabucketexp12'. The main area is titled 'lambdabucketexp12 Info'. Under the 'Objects (0) Info' section, there is a search bar and a table header with columns 'Name', 'Type', 'Last modified', 'Size', and 'Storage class'. A message states 'No objects' and 'You don't have any objects in this bucket.' At the bottom, there is a large 'Upload' button.

Upload a random saved image from your device onto the bucket by clicking on 'Add files' inside the bucket

The screenshot shows the 'Upload' interface for the 'lambdabucketexp12' bucket. The top navigation bar includes 'Services', 'Search', and 'N. Virginia'. The breadcrumb path shows 'Amazon S3 > Buckets > lambdabucketexp12 > Upload'. The main area is titled 'Upload Info'. It features a 'Drag and drop files and folders you want to upload here...' area, a 'Files and folders (1 Total, 6.6 KB)' table with one item 'images.jpg', and a 'Destination' section set to 's3://lambdabucketexp12'. At the bottom, there is a 'Close' button.

After uploading the image onto the bucket, this is the screen that will appear...

The screenshot shows the 'Upload: status' interface. The top navigation bar includes 'Services', 'Search', and 'N. Virginia'. The main area is titled 'Upload: status' and displays a summary table with one row: 'Destination s3://lambdabucketexp12' (Status: Succeeded, 1 file, 6.6 KB (100.00%)) and 'Failed 0 files, 0 B (0%)'. Below this is a 'Files and folders (1 Total, 6.6 KB)' table with one item 'images.jpg' (Status: Succeeded). At the bottom, there is a 'Close' button.

Step 3: Modifying Triggers inside Lambda configuration

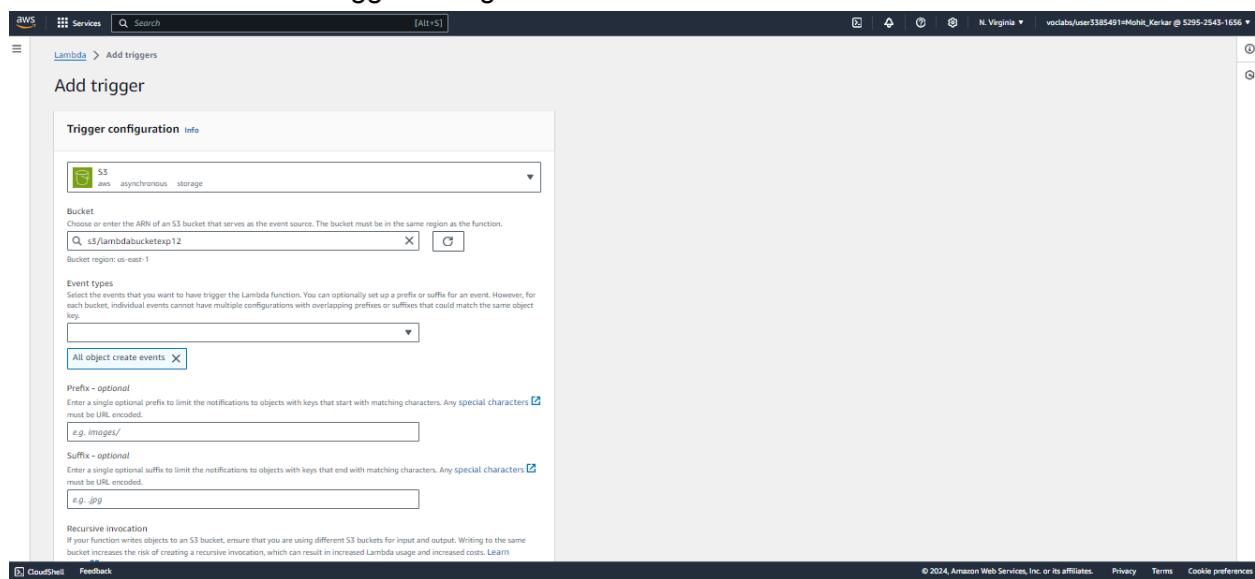
Go back to the lambda function that we created in the previous experiment

The screenshot shows the AWS Lambda console. In the top navigation bar, 'Services' is selected. The main search bar contains 'Search [Alt+S]'. The top right corner shows 'N. Virginia' and a user profile. The main content area is titled 'Lambda > Functions > mohit'. The 'Function overview' section is displayed, with the 'Code' tab being active. The central part of the screen shows a diagram with a single function node labeled 'mohit' and a 'Layers' section showing '(0)'. Below the diagram are buttons for '+ Add trigger' and '+ Add destination'. To the right, there are sections for 'Description', 'Last modified' (24 minutes ago), 'Function ARN' (arn:aws:lambda:us-east-1:529525431656:function:mohit), and 'Function URL' (Info). At the bottom, there are tabs for Code, Test, Monitor, Configuration, Aliases, and Versions.

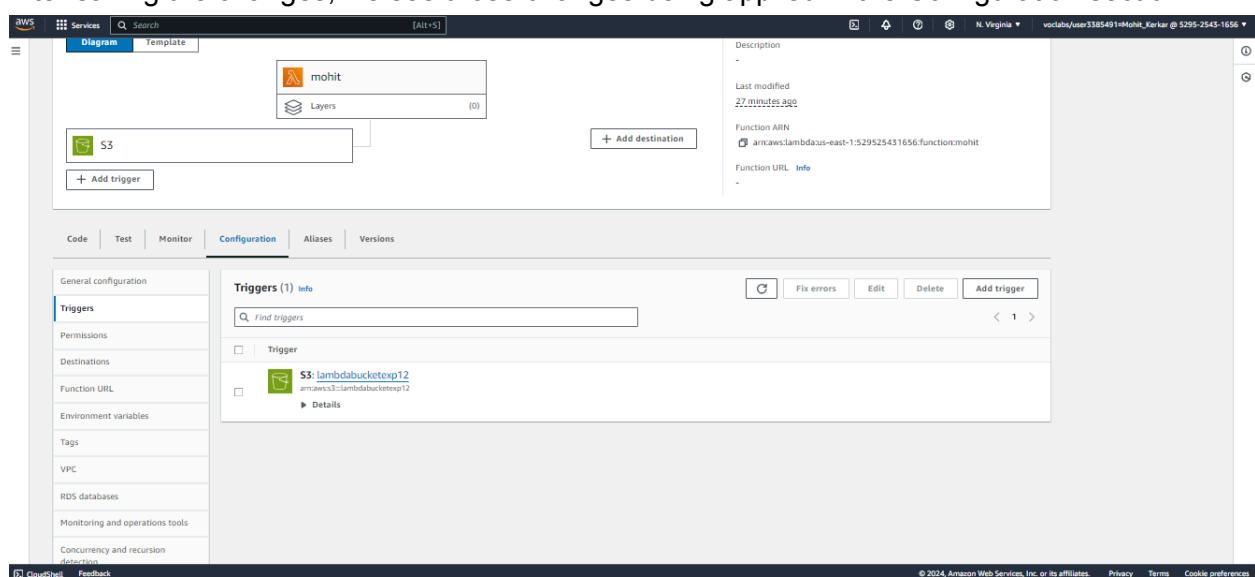
Go to the configuration section inside the function and select Triggers. This is what should appear on the screen...

The screenshot shows the AWS Lambda console. In the top navigation bar, 'Services' is selected. The main search bar contains 'Search [Alt+S]'. The top right corner shows 'N. Virginia' and a user profile. The main content area is titled 'Lambda > Functions > mohit'. The 'Configuration' tab is selected. The left sidebar shows 'General configuration' with 'Triggers' selected. The main content area shows the 'Triggers (0)' section with a search bar and an 'Add trigger' button. A message states 'No triggers are configured.' Navigation tabs at the bottom include Code, Test, Monitor, Configuration, Aliases, and Versions.

Select S3 bucket inside Trigger configuration to use the S3 bucket that was created earlier



After saving the changes, we see those changes being applied in the Configuration section



Step 4: Testing our code

In the code section, we are replace the default code with the following code:

```
export const handler = async (event) => {
  if (!event.Records || event.Records.length === 0) {
    console.error("No records found in the event.");
    return {
      statusCode: 400,
      body: JSON.stringify('No records found in the event')
    };
}
```

```

// Extract bucket name and object key from the event
const record = event.Records[0];
const bucketName = record.s3.bucket.name;
const objectKey = decodeURIComponent(record.s3.object.key.replace(/\+/g, ' ')); // Handle
encoded keys

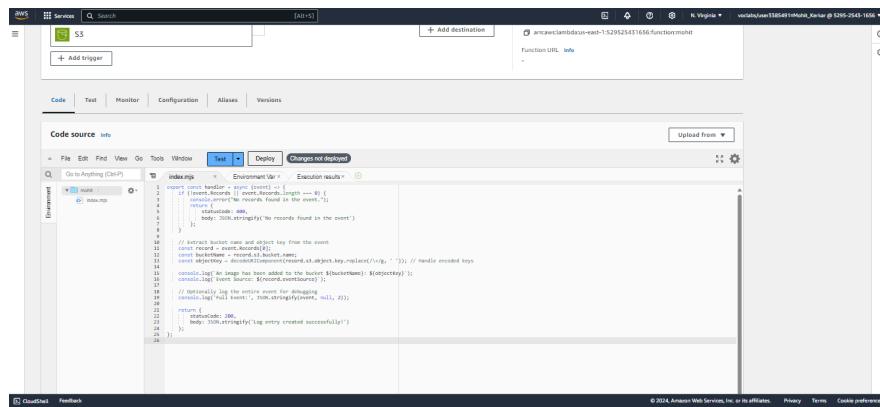
console.log(`An image has been added to the bucket ${bucketName}: ${objectKey}`);
console.log(`Event Source: ${record.eventSource}`);

// Optionally log the entire event for debugging
console.log('Full Event:', JSON.stringify(event, null, 2));

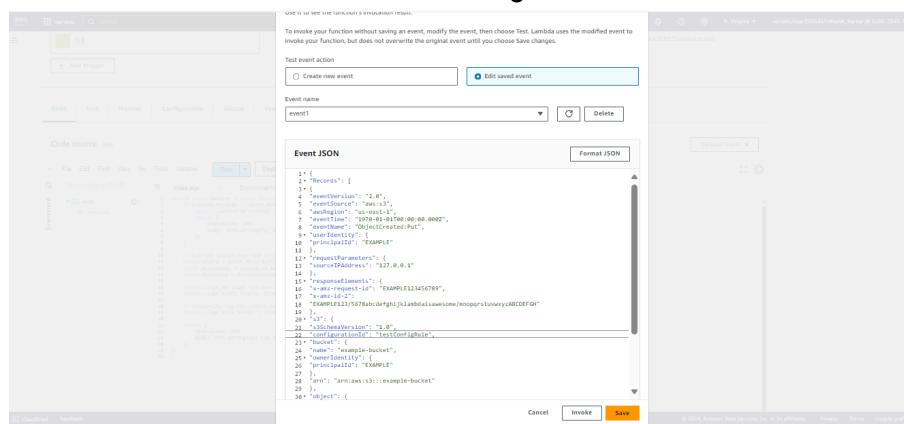
return {
  statusCode: 200,
  body: JSON.stringify('Log entry created successfully!')
};

}

```



For this particular code, we are supposed to edit the configurations for our test and save the event's JSON as we see in the following screenshot



Click on deploy

The screenshot shows the AWS Lambda console. In the top navigation bar, 'Services' is selected. Below it, 'Lambda' is chosen from a dropdown menu. The main area displays the 'mohit' function. Under the 'Code' tab, there is a 'Test' button highlighted in blue. The 'Execution results' tab is open, showing the code for the 'index.js' file. The code is as follows:

```

1  export const handler = async (event) => {
2    if (!event || !event.Records) {
3      console.error(`No records found in the event.`);
4      return;
5    }
6    const statusCode = 200;
7    const body = JSON.stringify(`No records found in the event.`);
8
9    // Extract bucket name and object key from the event
10   const record = event.Records[0];
11   const bucketName = record.s3.bucket.name;
12   const objectKey = decodeURIComponent(record.s3.object.key.replace(/\+/g, ' ')); // Handle encoded keys
13   const eventSource = record.eventSource;
14
15   console.log(`An image has been added to the bucket ${bucketName}: ${objectKey}`);
16   console.log(`Event source: ${eventSource}`);
17
18   // Optionally log the entire event for debugging
19   console.log(`Full Event: ${JSON.stringify(event)}`);
20
21   return {
22     statusCode,
23     body,
24   };
25 };
26
27 module.exports.handler = handler;
  
```

Click on test after deploying. This is the execution result that we see. Here, we see a line being mentioned inside the execution result.... 'An image has been added to the bucket'.

The screenshot shows the AWS Lambda console with the 'Execution results' tab selected. It displays the output of a recent execution. The status is 'Succeeded' with a duration of 35.00 ms. The memory usage was 64 MB. The execution ID is 24000000-0000-0000-0000-000000000000. The 'Response' field shows the JSON output: { "statusCode": 200, "body": "\u0022Log entry created successfully!\u0022" }. The 'Function Logs' section shows the log entries for the execution, including the message 'An image has been added to the bucket example-bucket: test/key'. The log entries are timestamped from 2024-08-07T09:49:28Z to 2024-08-07T09:49:28Z.

Step 5: Check logs

Search for CloudWatch. Click on logs -> logs groups

The screenshot shows the AWS CloudWatch service. In the left sidebar, 'Logs' is expanded, and 'Log groups' is selected. The main area shows a table titled 'Log groups (4)'. The table lists four log groups: '/aws/lambda/RedshiftEventSubscription', '/aws/lambda/RedshiftOverwatch', '/aws/lambda/RoleCreationFunction', and '/aws/lambda/mohit'. Each log group is listed with its type (Standard), configuration status (Configure), and retention settings (Never expire). There are also columns for Log class, Anomaly detection, Data protection, Sensitive data controls, and Metric filters.

In here, select the logs that we want to see

The screenshot shows the AWS CloudWatch Log Groups interface. The left sidebar is collapsed. The main area shows the log group details for `/aws/lambda/mohit`. It includes sections for Log class (Info), Standard, ARN (arn:aws:log:us-east-1:529525451656:log-group:/aws/lambda/mohit), Creation time (29 minutes ago), Retention (Never expire), and Metrics filters (0). On the right, there are sections for Stored bytes, Metric filters (0), Subscription filters (0), Contributor Insights rules, KMS key ID, Anomaly detection (Configure), Data protection, and Sensible data count. Below this, a tab bar has Log streams selected. Under Log streams, there are two entries: `2024/10/07/[SLATEST]bd1100987a1e4a57975c78ee40b323dc` (Last event time: 2024-10-07 09:49:20 (UTC)) and `2024/10/07/[SLATEST]3cc6eb1aa2cd4c61b0dde8f7d63941` (Last event time: 2024-10-07 09:24:15 (UTC)). A search bar at the top of the log stream list allows filtering by prefix.

Here, we see it saying 'An image has been added to bucket', which matches the execution results in the test that we performed on our code

The screenshot shows the AWS CloudWatch Log Events interface. The left sidebar is collapsed. The main area shows log events for the log stream `2024/10/07/[SLATEST]bd1100987a1e4a57975c78ee40b323dc`. A filter bar at the top allows searching for terms in the log events. The log events table has columns for Timestamp and Message. One event is highlighted: `2024-10-07T09:49:20,722Z START RequestId: 22400360-cf0f-46a8-82a4-2a409b0720bc Version: \$LATEST 2024-10-07T09:49:20,722Z 22400360-cf0f-46a8-82a4-2a409b0720bc INFO An image has been added to the bucket example-bucket: test/key`. Other events listed include `2024-10-07T09:49:20,733Z 22400360-cf0f-46a8-82a4-2a409b0720bc INFO Event Source: aws:s3` and `2024-10-07T09:49:20,739Z 22400360-cf0f-46a8-82a4-2a409b0720bc INFO Full Event: { "Records": [{ "eventVersion": "2.0", "eventSource": "aws:s3", "awsRegion": "us-east-1", "eventTime": "1970-01-01T00:00:00Z", "requestId": "22400360-cf0f-46a8-82a4-2a409b0720bc", "s3": { "size": 128, "objectKey": "test/key", "bucket": "example-bucket", "objectVersion": null } }] } 2024-10-07T09:49:20,758Z END RequestId: 22400360-cf0f-46a8-82a4-2a409b0720bc 2024-10-07T09:49:20,758Z REPORT RequestId: 22400360-cf0f-46a8-82a4-2a409b0720bc Duration: 35.09 ms Billed Duration: 36 ms Memory Size: 128 MB Max Memory Used: 64 MB Init Duration: 149.23 ms`.

Conclusion: Integrating AWS Lambda with S3 allows for real-time, automated processing of events such as file uploads. In this example, a Lambda function is configured to log a message whenever an image is added to a specific S3 bucket. This setup demonstrates the power and flexibility of serverless computing by automating tasks without requiring manual intervention or server management. By leveraging AWS Lambda, developers can efficiently handle event-driven workflows, reduce operational overhead, and quickly deploy scalable solutions that respond to specific actions within cloud environments.